INTRODUCTION

The present text includes the description of the basic constructions in the system PC Mizar, but the description is not complete. The text consists of four chapters and the annex containing a number of examples.

Chapter I discusses terminological issues and the symbolism used. Chapter II describes the fundamental constructions in Mizar, namely article and directives. Identifiers, reserved words and symbols, and numerals are discussed, too. Chapter III is concerned with formulas, and Chapter IV, with proofs of theorems.

The text is mainly concerned with the syntactics of Mizar. Elements of semantics, indispensable for the explanation of certain rules of proofs, are discussed in III.7 "Semantic correlates".

The text includes a number of examples (mainly from general topology), to be found both in the annex and in the main text. This should facilitate one both to learn Mizar and independently to write articles in that language.

The author is indebted to Dr A.Trybulec and to G.Bancerek for valuable suggestions and comments, very helpful in writing of the present text.

PC Mizar system is implemented by A.Trybulec and Cz. Byliński. Andrzej Trybulec is the author of the Mizar language.

I. CONVENTIONS

Every Mizar article is a sequence consisting of ASCII symbols (ASCII: a fixed code of signs arranged in a certain order) other than control signs, the sign No. 127 and No. 255. Fragments of Mizar articles presented in this text will, however, include signs not represented in the ASCII code (such as $\mathfrak{F}$, $\mathfrak{G}$, $\mathfrak{H}$). Those signs are used in order to increase the legibility of the text.

The table below lists the symbols not allowed in a Mizar article, which will be used in the present paper, and their analogous in the standard ASCII.

Moreover the text includes inscriptions of the form:

list–...

which will be termed lists, as well as other inscriptions consisting of words linked by the hyphen ”–”, e.g.,

segment–of–qualified–variables,
symbol–of–functor.

Hyphenation is intended to indicate that the words thus liked together form a certain whole.

Further, certain words will be written in **bold** type. They will be words reserved for Mizar, that is such whose meanings are rigorously determined by definition in the Mizar language. That typographical distinction is to draw the Reader's attention to then, and thus more easily to remember at least some of them. Note that the list of all reserved words and symbols will be found in the present text.

Symbol in this book	Representation in ASCII
$\mathfrak{F}$	F
$\mathfrak{G}$	G
$\mathfrak{H}$	H
$\subseteq$	c=
^{c}	‘
$\cup$	U
Ω	[234]
$\emptyset$	[237]
$\in$	[238]
$\cap$	[239]
$\leq$	[243]
$\neq$	<>

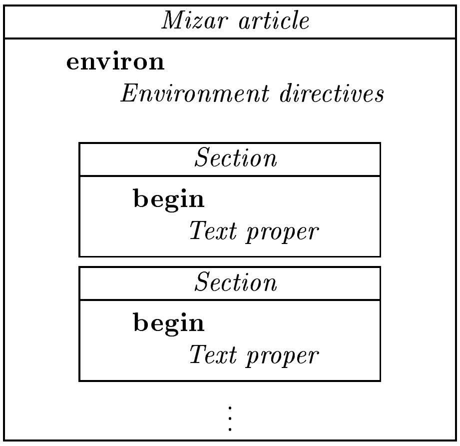
The following symbolism has been adopted:

$\mathfrak{G}$	– topological space,
$\mathfrak{H}$	– subspace of a topological space $\mathfrak{G}$,
$\mathfrak{F}$	– family of subsets of a topological space $\mathfrak{G}$,
A, B, G, P, Q	– subsets of a topological space $\mathfrak{G}$,
X, Y, M, N	– sets,
p, q	– points of a topological space $\mathfrak{G}$,
k, l, n	– natural numbers,
x	– arbitrary objects.

II. ARTICLE AND DIRECTIVES

A file with a Mizar text is termed a *Mizar article.* The name of the file may consist of not more than eight signs: letters, figures, underscorings (_), and the sign `'`, and may be neither a numeral nor a reserved Mizar word. Moreover such a file must have the extension `.miz`.

Here are some examples of names of a file: `W1.miz`, `''_'.miz`, `x.miz`. In view of the proof clarity the use as the name of a file of the inscription `''_'` or any other equally little legible inscription is not recommended. A *Mizar article* consist of two parts: *environment directives* and the sequence of *sections*, which must be separated from one another by the word **begin.** The environment directives must be preceded by the word **environ,** which opens every *Mizar article.*



The *text proper* may include, among other things, proofs of theorems, definitions with conditions of correctness, proofs of schemata. In order to write a correct non-empty *text proper* one needs the environment which for the person who writes that article can be organized by *environment directives.* They include items of information indispensable for the correct reading by Mizar of the *text proper*, and are the basis for proofs. To put it more rigorously, *environment directives* refer to a data base and thus indicate which elements in the existing library are used in a given article.

The *environment directives* include:

vocabulary α ;
signature β ;
definitions γ ;
theorems δ ;
schemes ε ;

where

α – the name of a vocabulary file (e.g., `TOPCON`, `ANAL`),
β – the name of a signature file (e.g., `TOPS_1`, `PRE_TOPC`, `SUBSET_1`),
γ – the name of a definition file (e.g., `TARSKI`, `BOOLE`),
δ – the name of a theorem file (e.g., `CONNSP_1`, `REAL_1`),
ε – the name of a schema file (e.g., `NAT_1`).

II.1. Vocabulary directive

The directive

vocabulary α ;

is termed *vocabulary directive*, and the remaining ones, *data base directives*. Every *Mizar article* consist of a certain numbers of symbols. Some of then are introduced automatically (hidden symbols), while the remaining ones are introduced by reference to *vocabulary directives*. Hence vocabularies are needed. A vocabulary consist of a file with the extension .voc. That file contains the list of symbols with their qualifiers and indicates the biding strength of the symbols of functors. For instance, the file TOPCON.voc, which forms the vocabulary, is as follows:

TOPCON.voc
OCl 128
OFr 128
Oskl 128
Ucarrier
Utopology
GTopStruct
Ris_open
Ris_closed
Ris_open_closed
Rare_separated
Ris_continuous
Rare_joined
Ris_a_component_of
Ris_a_cover_of
MTopSpace
MPoint
MSubSpace
Mmap

In its leftmost column it contains qualifiers, and beginning with the next column to the right until the space it contains symbols. Qualifiers in a sense characterize symbols. For instance the qualifier O indicates that the symbol next to it is a symbol of a functor, while the qualifier R indicates that the symbol next to its is a symbol of a predicate. Thus the symbols

Cl, Fr, skl

are symbols of functors, while the symbols

is_open, is_closed, is_open_closed, are_separated, is_continuous,
are_joined, is_a_component_of, is_a_cover_of

are symbols of predicates.

The symbols Cl, Fr, skl denote, respectively the operations of: closure, boundary of a subset of a topological space, and component of a point of a topological space.

The symbols is_open, is_closed, is_open_closed are used to denote predicates defined for subsets of a topological space and indicating, respectively, that is given set is open, closed, open-closed. The symbol is_continuous is used to denote the property of being a continuous mapping of topological spaces. The symbol are_separated denotes the relationship between subsets of a topological space which says that they belong to one and the same component. The symbol is_a_component_of denotes two predicates: one says that a subset of a topological space is the maximal compact set (component) in that topological space, and the other says that it is a component in another subset of a topological space. The symbol `is_a_cover_of` denotes the property of being a cover of a topological space.

Other qualifiers occurring in the file `TOPCON.voc` are:

`G` – qualifier of the symbol of structure,
`U` – qualifier of the symbol of selector,
`M` – qualifier of the symbol of mode.

The symbol `TopStruct` is used to denote the structure of a topological space, and the symbols `topology` and `carrier` denote, respectively, the topology and the carrier of a topological space. The symbols of modes, i.e., the inscriptions `TopSpace`, `SubSpace`, `Point`, `map`, are used to denote, respectively, topological space, topological subspace, point of a topological space, and a mapping between topological space.

Qualifiers of functor brackets may also occur:

`K` – left functor bracket,
`L` – right functor bracket.

The identifier of schema is not introduced into vocabulary.

Here is the file `ANAL.voc` which contains the symbols of functor brackets used in defining the absolute value:

ANAL.voc
K |.
L .|
Osgn

The inscriptions:

$< *, * >$, <:, :>

are other examples of functor brackets. These are used to denote, respectively, finite sequences and functions which are pairs of functions. There is a pair of functor brackets whose symbols are in the file `HIDDEN.voc`. **That file is joined automatically to every article.**

Moreover, the vocabulary file indicates the binding strength or priority. This applies only to the symbols of functors. The binding strength of a given functor is indicated by the number next to its symbol.

Remark: The number characterizing the priority of a given functor must be separated from the symbol of that functor by at least one space.

There can be no space between the qualifier and its corresponding vocabulary symbol.

Self-evidently, the symbol of a functor binds more strongly if its number is greater. The priority of a given functor may be characterized by any natural number in the interval <0;255>.

All symbols of functors given in the vocabulary Topcon have the priority 128. Some symbols of functors have no number characterizing priority, but this is not say that a given symbol has no priority. That priority is given and amounts to 64. This is the standard priority.

For instance, the binding force of the symbol of the functor `sgn` to be found in the vocabulary Anal presented above is not given.

Remark: The binding force of the symbols of predicates, which always bind more weakly than do the symbols of functors, is not given.

The concept of *binding force* of the symbols of functors is linked to the sequence in which the operations in a given formula are performed. Consider, for instance, two formulas:

$$\texttt{Cl P}^c \quad \text{and} \quad \Omega\mathfrak{G} \cap \mathrm{Q}.$$

Since the binding force of the symbol ^{c} is greater (it amounts to 150) than that of the symbol `Cl` (128), the inscription `Cl` $\texttt{P}^c$ is interpreted as the closure of the complement of the set `P`, that is, in the same way as the inscription `Cl` $(\texttt{P}^c)$. It is likewise in the second case. The priority of the symbol Ω is 128, and that of them symbol ∩ is the standard one, i.e., 64. Hence the inscription $\Omega\mathfrak{G} \cap \texttt{Q}$ is interpreted in the same was as the inscription $(\Omega\mathfrak{G}) \cap \texttt{Q}$. The acknowledge of the priority of at least some symbols may be used in articles in order to avoid superfluous brackets.

II.2. Identifiers

Inscription which include: ASCII control signs (i.e., signs which have ordinal numbers from 0 to 31), space (sign with the number 32), and the signs with the numbers 127 and 255, cannot be vocabulary symbols.

Mizar articles include inscription termed identifiers. What sort of an inscription on identifier is? Now *identifier* is any non-empty sequence of certain signs. Those signs may be letters, figures, the symbol of underscoring (_), and apostrophe ('), but not reserved words, not reserved symbols of Mizar nor numerals (see II.5.). The length of an inscription which is an identifier should not exceed sixteen signs because otherwise such an inscription which is an identifier may be a vocabulary symbol, but not conversely.

Identifiers are used to denote:

a) private functors and predicates,

b) variables,

c) labels.

Hence we may speak about identifiers of variables, identifiers of private functors and predicates (if it is not a private functor or predicate then we speak about a symbol), etc.

By way of example we shall specify the identifiers in the file `Z1.lst` included in the annex. They are as follows:

- identifiers of variables:

  `T`, `P`,

- identifiers of labels:

  `Z1`, `Z2`.

The identifiers of labels are examples of references. References make it possible to refer to sentences which have been earlier assumed or substantiated.

$$\text{references}\begin{cases}\text{local} \;-\; \textit{identifiers of labels}\\[2ex] \text{library} \;-\; \textit{file symbol} \;:\; \begin{cases} \textit{number} \\[1ex] \texttt{def}\ \textit{number}\end{cases}\end{cases}$$

Examples of local references have been given above.

Library references are exemplified by the inscriptions:
`TOPS_1:28`, `BOOLE:1`, `TARSKI:4`, `REAL_1:5`, `SUBSET_1:14`, `PRE_TOPC:34`.

A library reference results in the reference to a definite theorem to be found in the Mizar library. For instance, the library reference `TOPS_1:28` results in the references to the theorem No.28 recorded in the file `TOPS_1.abs`. On the contrary, local references apply to sentences in a given article and unlike library references may be freely assigned to sentences.

Sentences are assigned labels so that one can refer to them in a later part of the text. As between signature directives (see III.1) the phenomenon of overridding may hold between identifiers of labels.

DISCRIMINANTS OF IDENTIFIERS OF NAME OF FILES:

▹ An identifier which is a name of a file consists maximally of eight signs.
▹ An identifier may be formed of:

letters, figures, the sign of underscoring (_) and apostrophe (').

▹ In an identifier capital letters and lower-case letters are treated as identical. For instance, the inscriptions `row`, `Row`, and `ROW` are one and the same name of a file.

The adopted convention is that names of files are always in capital letter.

An inscription which is an identifier has a close connection with those vocabulary files which have been used in the environment. The point is that the symbols in those files cannot be identifiers. Should we disregard that errors would be reported as in the example `Z2.lst` in the annex. They resulted from the use of the inscription `Fr` as an identifier of a variable. Note that `Fr` is the symbol of a functor included in the vocabulary Topcon, and that vocabulary is joined to the environment. Hence, in accordance with what has been said earlier, it was not allowed to use the inscription `Fr` as an identifier of a variable.

Remark: The person who writes has large freedom in constructing identifiers, and this is why attention is drawn to the fact that the inscriptions which function as identifiers should be as legible as possible because that contributes to both the clarity of that article and its aesthetic appearance.

II.3. Hidden vocabulary

HIDDEN.voc
MAny MElement MDOMAIN MSubset MSET_DOMAIN MSUBDOMAIN MReal MNat K[: L:] Obool 128 OREAL 255 ONAT 255 O+ 32 O· R<> R∈ R≤ R≥ R< R>

The symbols of functors `+` and · are used to define, respectively, the addition and the multiplication of terms, whose type is expanded to the type `Element of REAL`. The symbols $\in$ and $\leq$ are used to denote, respectively, the relation of membership and the relation of order. The correct use of the predicate symbolized by $\in$ consist in that the type of its left argument must be expandable to the type `Any`, and that of the right argument, to the type `set`. The predicate symbolized by $\leq$ is defined to objects whose type is expandable to the type `Element of REAL`. The inscription `bool` is used as the symbol of the functor which denotes the family of all subsets of a certain set. The symbols of the functor brackets to be found in the Hidden vocabulary are adopted to denote of Cartesian product of sets. The remaining symbols in that vocabulary will be discussed in the section dedicated to types.

II.4. Data base directives

The *signature directive* will be discussed first. The directive

signature β ;

joins automatically the files: β.`sgn`, β.`nfr` and β.`typ`. They contain information about the way in which the symbols introduced in the vocabularies may be used. For instance, the file β.`sgn` lists the types of arguments of the objects defined and patterns of definitions. The file β.`nfr` contains the descriptions of the formats of the objects defined (functors, predicates, modes). Formats for schemata are not given. A format offers information about the number of arguments. One and the same symbol may have several formats. For instance, the symbol $\emptyset$ is used in the article `BOOLE` in the format of 0–0 arguments to denote the empty set (zero left arguments and zero right arguments – see table in III.1.), while in the article `PRE_TOPC` in the format of 0–1 arguments, to denote the least element of the family of open sets of a given topological space (zero left arguments and one right arguments – see table in II.1.). In both cases the priority is the same because it pertains to the symbol of a functor. The content β.`typ` contains the types of the result of a functor and the type of the expansion of a mode.

For instance, the joining to the environment of the directive

signature `BOOL;`

results in the symbol $\cap$ (to be found in the vocabulary Boole – file `BOOLE.voc`) being correctly usable for denoting the two-argument operation of intersection, where the left and right argument are sets. Moreover, the results of the operation $\cap$ is a set, too. The context in which the symbol $\cap$ is interpreted here follows from the definition of intersection, given in the article Boole (where `X, Y` are identifiers of sets).

```
definition
    let X,Y;
    func X ∩ Y -> set means
    x ∈ it iff x ∈ X & x ∈ Y;
end;
```

Let use analyse the part of the definition `X ∩ Y -> set`. It follows from `X ∩ Y` that the operation $\cap$ is a two-argument one (the sets `X` and `Y` being the arguments). The symbol `set` after the symbol `->` informs one that the results of the operation $\cap$ is a set.

The directive

signature `BOOL;`

makes accessible all definitions (which are not everridden) to be found in the article Boole. This applies, among other things, to the definition of the operation of intersection denoted by the symbol $\cap$ (number of arguments, types of arguments, type of result of the operation).

But the operation denoted by the symbol $\cap$ may be also interpreted otherwise. The article `PRE_TOPC` includes a redefinition of the symbol $\cap$:

definition

```
let 𝔊, P, Q;
redefine
func P ∩ Q -> Subset of 𝔊;
end;
```

where P, Q are variable identifiers of subsets of the topological space $\mathfrak{G}$.

If the directive

signature PRE_TOPC;

is joined to the environment, then the symbol $\cap$ will be used to denote the two-argument operation of intersection where both the left and the right argument is a subset of the topological space $\mathfrak{G}$. Moreover, the result of the operation $\cap$ is also a subset of the topological space $\mathfrak{G}$.

The application of a signature directive should in that case be included in the environment?

The operations denoted by the symbol Ω, ' , $\cap$, $\cup$, $\setminus$ for subsets of the topological space $\mathfrak{G}$ have been defined in the article PRE_TOPC. In the case of the first two symbols we have to do with definitions, in the case of the remaining ones, with redefinitions. Since the identifiers of variables which are arguments of the operations denoted by the symbols indicated above are, in the exercise, reserved for subsets of the topological space $\mathfrak{G}$, the directive

signature PRE_TOPC;

should be joined to the environment. There will be also the information about the mode TopSpace. The mode with the symbol Subset is to be found in the vocabulary HIDDEN, automatically joined to every article, and hence cannot occur among environment directives. Moreover, the information about the use of the symbols to be found there are automatically used by the processor of PC Mizar, that is without the indication of the corresponding signature directives

The examples Z4.lst and Z5.lst in the annex illustrate errors due to a lack of the proper signature directive.

Remark: The order in which signature directives are specified may be importance. Such is the case in the redefinitions of one and the same symbol. The valid redefinition is always that of the last signature specified in the environment. If that order is erroneous, then the objects defined in a given will be everridden.

The example below shows the overridding of the operation of intersection defined in **signature** PRE_TOPC; . Places where the error No. 103 is reported are indicated.

```
environ
   vocabulary SUB_OP;
   vocabulary BOOLE;
   vocabulary TOPCON;
   signature PRE_TOPC;
   signature BOOLE;
   theorems BOOLE;
   theorems TOPS_1;
begin
   reserve 𝔊 for TopSpace,P,Q for Subset of 𝔊;
   (P ∩ Q)^c = P^c ∪ Q^c
         *103
     proof
        (P ∩ Q)^c = Ω𝔊 \ (P ∩ Q) by TOPS_1:5
              *103
```

```
.= (ΩG \ P) ∪ (ΩG \ Q) by BOOLE:86
.= P^c ∪ (ΩG \ Q) by TOPS_1:5
.= P^c ∪ Q^c by TOPS_1:5;
hence thesis;
end;
```

(Consider the example Z6 in the annex.)

Since the last signature directive is the directive **signature** `BOOLE;`, the operation denoted by the symbol ∩ has been used in the sense defined in the article BOOLE (the subset of a topological space are sets, too). In accordance with that definition the results of the operation of intersection is a set.
Hence the intersection `P ∩ Q` is a set. But the closure operation is defined only for subsets of a fixed set. That is why the expression `(P ∩ Q)`^{c} is followed by the indication of an error.

The overridding of the directive **signature** `PRE_TOPC;` can be avoided if the order of the signatures occurring in the example under consideration is changes (as has been done in the example `Z7.lst`).

Proofs are sometimes carried out by the method of definitional expansion. In such a case the directive.

definitions γ ;

should be joined to the environment.

Proving by definitional expansion will be illustrated by an example. The proof of the theorem is given below:

For any sets X,Y we have: X ∩ Y ⊆ Y.
The proof (not in the Mizar notation) is as follows:

Let a be an arbitrary but fixed and such that a ∈ X ∩ Y.
1) a ∈ X ∩ Y (assumption of the proof);
2) a ∈ X ∧ a ∈ Y (1, definition of the intersection of sets);
3) a ∈ Y (2, the law of the omission of conjunction).

It follows from the arbitrariness of the choice of a and the definitional expansion that X ⊆ Y.

X ⊆ Y ⇔ ∀a (a ∈ X ⇒ a ∈ Y) – definitional expansion of inclusion.

When proving in Mizar the above theorem by reference to definitional expansion one should join to the environment the directive

definitions `TARSKI;`

because in the article TARSKI there is the definition of inclusion which is as follows:

pred `X ⊆ Y` **means** `x ∈ X` **implies** `x ∈ Y;`

And here is the redefinition of the quality of sets, to be found in the article BOOLE:

pred `X = Y` **means** `X ⊆ Y & Y ⊆ X;`

In example one in the file `art.lst` the theorem has been proved in two ways. In both cases use has been made of the definitional expansion of inclusion and the definitional expansion of the equality of sets. That is why the *environment directives* include two *definition directives*:

definitions `TARSKI;` and **definitions** `BOOLE;`.

The *definition directive*

definitions γ ;

automatically joins the file γ`.def`, which includes the definitienses of the objects defined (definiens – the expression which occurs in a definition of a functor, a predicate, a mode, a attribute after the word `means`).

The *theorem directive*:

theorems δ ;

allows one to make use of the theorems in file δ`.miz`. The writing of that directive results in the automatic joining of the file δ`.the`, which includes the contents of the theorems in a given article.

The directive

schemes ε ;

allows one, through the automatic joining of the file ε`.sch`, containing contents of the schemata in the file ε`.miz`, to use the schemata in that file.

For instance, the induction schema is to be found in the article NAT_1. Hence, in order to use it one has to join to the environment the directive **schemes** `NAT_1;`, that is, to insert it between the word **environ** and the word **begin**.

If the text requires several vocabularies one has to repeat the directive

vocabulary *name–of–file* ;

with the corresponding names of vocabulary files. In the case of the remaining directives one has to proceed analogically.

Remark: The repetition of a directive with the same name of the file yields an error. But it is not so if a directive superfluous for a given article is added, as in the example `Z7.lst`*, where the directive* **signature** `BOOLE;` *is superfluous.*

BRIEF DESCRIPTION OF THE ORGANIZATION OF THE MIZAR DATA BASE

In the main mizar directory `\MIZAR` there are two subdirectories:

`\DICT` – intended for vocabulary files (files with the extension `.voc`),

`\PREL` – intended for library files formed by the program called LIBRARIAN. Those files are formed automatically and have the extensions :

`.sgn`, `.nfr`, `.typ`, `.def`, `.the`, `.sch`.

They form the *Data Base.*

The presence of those subdirectories in the disc memory of the computer is necessary because it is from them that the Mizar processor draws information which make it possible to write Mizar articles. The subdirectory `\ABSTR` is often formed additionally.

`\ABSTR` – intended for library files which are obtained from mizar articles after their special processing. Files in that subdirectory are termed *abstracts* and have the extension `.abs`. The abstracts contain in their main part contents of theorem and definitions, and schemata. They do not contain proofs. The theorem in the file #`.abs` (where # stands for the name of a given article) are numbered. Every theorem in the file #`.abs` is preceded by a headline in the form:

`::  # :` *number–of–theorem*

The subdirectory `\ABSTR` plays only an auxiliary role for the user. When perusing the files in that subdirectory one can learn what has already been proved in Mizar. Moreover, if one wants, in the proof of a certain sentence, to refer to a theorem from a file in the *Main Mizar Library*, then one can read the name of that file and the number of the theorem and refer to them in the proper place. But it is not necessary for the subdirectory `\ABSTR` to be recorded in the computer memory. The Mizar processor uses only the information given in the files from the subdirectories `\DICT` and `\PREL`.

II.5. Words reserved for Mizar. Reserved symbols. Numerals

The words reserved for Mizar are drawn from the English language. They are inscriptions whose meanings are defined by the definition of the Mizar language. For instance, **environ** is a word reserved for Mizar. It opens every Mizar article. That word may occur in article only once and only at the beginning. The use of **environ** in another context yields an error. Other reserved words also have their precisely defined meanings.

It words adding here there are also symbols reserved for Mizar, whose meanings, too, are fixed in advance. They include:

```
=    &    ,    ;    :    (    )    [    ]    {    }    ->
.=   <>   (#   #)
$1   $2   $3   $4   $5   $5   $6   $7   $8
```

The numerals include zero (`0`) and any finite sequence of figures not beginning with zero. The Mizar processor makes it possible to use numerals in the interval <0;255>. For instance, the inscriptions `00`, `0103` are not numerals.

The list of words reserved for Mizar:

aggregate	**and**	**antonym**
as	**associativity**	**assume**
attr	**be**	**begin**
being	**by**	**canceled**
case	**cases**	**cluster**
coherence	**compatibility**	**consider**
consistency	**contradiction**	**correctness**
def	**define**	**definition**
definitions	**end**	**environ**
ex	**exactly**	**existence**
for	**from**	**func**
given	**hence**	**hereby**
holds	**if**	**iff**
implies	**irreflexivity**	**is**
it	**let**	**means**
mode	**non**	**not**
now	**of**	**or**
otherwise	**over**	**per**
pred	**prefix**	**proof**
provided	**qua**	**reconsider**
redefine	**reflexivity**	**reserve**
scheme	**schemes**	**selector**
set	**signature**	**st**
struct	**such**	**symmetry**
synonym	**take**	**that**
the	**then**	**theorem**
theorems	**thesis**	**thus**
uniqueness	**vocabulary**	**where**

III. TERMS AND FORMULAS

The skill of constructing sentences in Mizar is a necessary condition if one is to write correctly a Mizar article. This is why in the present chapter we shall discuss the basic elements connected with the Mizar sentence. We mean terms and formulas. Let us begin with terms.

III.1. Terms

The set of terms is the least set which satisfies the following conditions:
a) variables and constants are terms;
b) if t_1, ..., t_p are terms and F is a symbol of a functor of p arguments, then $F(t_1,...,t_p)$ is a term.

But in the Mizar language the concept of term is interpreted more broadly. Terms in Mizar are inscriptions which are listed below under given categories.

(1). IDENTIFIERS OF VARIABLES ARE TERMS

For instance, they may be such inscriptions as: `P`, `TS`, `q`.

(2). NUMERALS ARE TERMS

For instance: `2`, `178`, `77`.

(3). THE EXPRESSION IN THE FORM:

list–of–leftside–arguments *symbol–of–functor* *list–of–rightside arguments*

is a term.

The functor symbol must be in the vocabulary. The inscriptions in the vocabulary are called symbols, and this is why, when speaking about symbols of functors, predicates, etc., we shall mean symbols of those functors predicates, etc., which are to be found in a certain vocabulary.

The number of arguments in the list of arguments (both leftside and rightside ones) may equal zero. Such is the case of the functor ∅. Moreover there may be cases in which the list of the leftside arguments equals zero, or that of the rightside arguments equals zero. Examples will be given below.

Consider the following symbols of functors which are used in articles pertaining to topological spaces: ∅, ∪, ∩, ‘, `Cl`, `Int`, `Fr`

and other symbols not occurring here. The table below shows the number of arguments of those terms

Functor symbol	Term	Number of left-side arguments	Number of right-side arguments
∅	∅	0	0
∅	∅𝔊	0	1
∪	P ∪ Q	1	1
∩	P ∩ Q	1	1
^{c}	P^c	1	0
Int	Int P	0	1
Cl	Cl P	0	1
Fr	Fr Q	0	1
FinUnion	FinUnion(B,f)	0	2
PLANE	PLANE(A,B,C)	0	3
All	All(x,y,z,H)	0	4
.	o.(a,b)	1	2
*	D*	1	0

If the list of (both leftside and rightside) arguments consists of at least two arguments, then such a list of arguments must be placed in brackets `(` and `)`, as has been done in the corresponding terms in the table above.

(4). EXPRESSION IN THE FORM:

leftside–functor–bracket *non–zero–list–of–terms* *rightside–functor–bracket*

are terms. A list–of–terms is a finite sequence of terms separated by commas.

Examples:

- Cartesian product of sets:
  `[:M1,M2:]` , `[:M1,M2,M3:]` , `[:M1,M2,M3,M4:]`
- absolute value of numbers `a` and `a - b`:
  `|.a.|`, `|.a - b.|`
- finite sequences of the length one and two, respectively:
  `<*k*>` , `<*k,l*>`
- function which is a pair of functions `f` and `g`:
  `<:f,g:>`

These are not all functor brackets, because the author of an article may introduce in the vocabulary ever new symbols for them.

Remark: Functor brackets must be used in pairs. In every pair brackets of the same kind should occur.

For instance, if in an expression the inscription `[` *is used as a leftside functor bracket, then the inscription* `]` *must be the rightside functor bracket in that expression.*

A pair of functor brackets between which there is no term is not a term.

Moreover there are brackets of two types which may be treated as functor of special kinds. They are:

`[, ]` and `{, }`

They can be used to construct terms of the following forms:

`[` *list–of terms* `]`, or `{` *list–of–terms* `}`.

Examples:

- ordered pairs, triples, and quadruples
  `[x,y]`, `[x,y,z]`, `[x,y,z,v]`
- singleton {x}, pair {x,y} and further finite sets up to those of eight elements:
  `{x1,x2,x3}`,
  `{x1,x2,x3,x4}`,
  `{x1,x2,x3,x4,x5}`,
  `{x1,x2,x3,x4,x5,x6}`,
  `{x1,x2,x3,x4,x5,x6,x7}`,
  `{x1,x2,x3,x4,x5,x6,x7,x8}`.

(5). AN INSCRIPTION IN THE FORM

`{` *term* `:` *formula* `}`

is a term. Such terms are called *Frænkel's operators.* As an example we may quote the following expression:

`{x : x <= 8}`, ($\{x : x \leq 8\}$)

where `x` is an identifier of a variable, reserved for the type `Real`.

Formulas will be discussed later (see the next section).

The types of the free variables occurring in the term now under consideration must expand to the type expanding to the type of the form `Element of` ⌈`DOMAIN`⌉. The inscription ⌈`DOMAIN`⌉ denotes any object of the type expanding to the type `DOMAIN`.

In the case of a *Frænkel operator* the types of the variables which occur in it may be given by writing out their type after the term. In such a case the *Frænkel operator* has the form:

{ *term* **where** *identifiers–of–variables* **is** *type* : *formula* }

If in a *Frænkel operator* there occur variables of more than one type, then the expression between **where** and : may be repeated the corresponding number of times separated by colon. The types given in such a formula refer to that formula only. For instance, if in the construction *reservation-of-variables* the identifier `x` were reserved for the type `Real` while in a *Frænkel operator* its type were changed into `Nat`, then in the further part of the article, in the formulas containing the identifier `x` but such in which its type would not be indicated, it would have the type assigned to it in the reservation of variables, that is `Real`.

As an example illustrating the *Frænkel operator* we may use Theorem 64 from the abstract `TREES_1`:

```
p ∈ T implies ⌈T,p,T1⌋ = {t1 where t1 is Element of
      T: not p is_a_proper_prefix_of t1}
            ∪{p^s where s is Element of T1:  s=s}
```

The identifier `p` has the type `Finsequence`, while the identifiers `T` and `T1` have the type `Tree`.

The symbols { }, `[ ]` are homonymous, which is to say that their meaning varies according to the context in which they are used. For instance, the brackets {, } under (4) above were used to denote sets of n-tuples where n $\leq$ 8, and under (5) the same brackets are used to denote a *Frænkel operator*. The symbols `[`, `]` are used to denote ordered pairs (triples, quadruples) as under (4) and as brackets in private predicates, e.g., `P[x]`.

The word `set` is homonymous, too. On the one hand, it is a type in Mizar (see II.4.), on the other, it occurs in the constructions `set` ... `=` ... and `set of` ..., in which it plays an entirely different role.

(6). AN EXPRESSION IN THE FORM

identifier–of–functor (*list–of–terms*)

is a term. Terms of this kinds are to be found, among other things, in definitions of local functors.

Here are two definitions of local functors, in which the identifiers `x`, `y`, `z` have the type `Element` **of** `RATIONAL`. The terms under consideration are:

`MULT(set,set)` and `UZUP(set)`

```
▷  func MULT (set,set) =
     {x·y:  x ∈ $1 & y ∈ $2 & 0 ≤ x & 0 ≤ y} ∪ {z:  z <0}
▷  func UZUP (set) = {-x:  not x ∈ $1}
```

(7). PARAMETERS OF A LOCAL DEFINITION

that is the symbols:

`$1`, `$2`, `$3`, `$4`, `$5`, `$6`, `$7`, `$8`

are terms.

Parameters of a local definition may be used in local definitions only.

(8). **it** IS A TERM

it may be used only in the definienses of functors, where it stands for the value of the functor, and in the definienses of modes, where it stands for that element of the mode which is given as an example.

The definiens of a functor is an expression which in the definition of that functor follows the word **means**. It is the same, *mutatis mutandis*, in the case of the definiens of a mode.

The term **it** occurs, for instance, in the definition of the functor which has the symbol `Int`, as given below.

definition
  let `P`, `𝔊`;
  func `Int P -> Subset` **of** `𝔊` **means it** `= (Cl` $(P^c))^c$;
end;

(9). AN EXPRESSION IN THE FORM
the *symbol–of–selector* **of** *term*

is a term.

The term in such a form is called *selector term.*

The type of the term which follows `of` must expand to a structure in the definition of which there occurs the symbol of selector used in the expression

the *symbol–of–selector* **of** *term*

In the structures of topological space, introduced in the article `PRE_TOPC`, there occur the following symbols of selectors:

`carrier`
`topology`

Examples of terms:

the `topology` **of** 𝔊, **the** `carrier` **of** 𝔊

(10). AN EXPRESSION IN THE FORM
the *symbol–of–selector*

is a term.

Terms of this kind may occur only in patterns of structures, and that only if the symbol of selector has been introduced earlier just in that pattern. As an example we may take the term

the `carrier`

occurring in the pattern of the structure of topological space `TopStruct`, presented below.

≪ `carrier -> DOMAIN, topology -> Subset_Family` **of the** `carrier` ≫

(11). AN INSCRIPTION IN THE FORM
symbol–of–structure ≪ *list–of–terms* ≫

is a term.

Terms in that form are called *aggregates of structures.* Since the symbols ≪, ≫ have no representation of their own in the standard ASCII the symbols `(#`, `#)` have been introduced and may be used alternately. Instead of ≪ and ≫ one may use, respectively, `(#` and `#)`, but not ≪ and `#)` or `(#` and ≫.

As an example one can give a definite structure (but not a pattern of a structure) such as that below:

`TopStruct` ≪ `REAL, RealTop` ≫

In the above structure it is `REAL` which is the carrier. The constant `REAL` expands to the type `DOMAIN`, which is necessary in view of the definition of the structure `TopStruct`. `RealTop` (topology) has the type `Subset_Family` **of** `REAL`, which is required by the definition of the structure `TopStruct`.

(12). AN INSCRIPTION IN THE FORM
term **qua** *type*

is a term.

It is a called a *qualified term.*

For instance, `P` **qua** `Subset` **of the** `carrier` **of** 𝔊.

The identifier `P` in the reservation is reserved for the type `Subset` **of** 𝔊, but in the term above its type has been as it were expanded to the type `Subset` **of the** `carrier` **of** 𝔊.

Remark: The word **qua** *only expands a type, it cannot narrow it down.*

(13). A TERM

which is in the brackets `(` and `)` is also a term.

Here are some examples with terms in brackets:

`Cl (P ∪ Q)`, `Fr (P ∪ Q)`, `(P \ Q)`^{c},

`(k + 1)`, `(n + 1)`.

Other examples will be given, among other things, when formulas are discussed.

III.2. Types

An identifier of a variable must have a type assigned to it.

In a Mizar article it is not allowed to use identifiers of variables whose types are unknown. The type of a given identifier may be fixed locally, that is given in the place where it occurs, or fixed globally by reservation (see below). Some modes are hidden in the Mizar language. The remaining ones must be identified. In order to construct a type we must define the appropriate mode. The symbol of a mode must be included in the vocabulary. Below we present types which use symbols of modes from the vocabulary HIDDEN (which is automatically joined to every Mizar article). Here they are:

`Any`, `set`, `Element` **of** `X`, `DOMAIN`,
`Subset` **of** `D`, `SUBDOMAIN` **of** `D`,
`Real`, `Nat`.
(where `X` has the type `set`).

The use of these types does not require from the author of a Mizar article the inclusion in the environment of any vocabulary or signature because the signature and the vocabulary required are joined automatically.

Remark: When one writes the types the important point is that their symbols be written precisely in the form in which they are to be found in the vocabulary. For instance, if one wants to reserve the identifier `K` *for the type* `DOMAIN`*, then the reservation should be in the form:*

reserve `K` **for** `DOMAIN;`

and not, for instance:

reserve `K` **for** `domain;`

Two examples more:

One should write:

`Element` **of** `REAL` and not `Element` **of** `real`,

and likewise

`Element` **of** `NAT` and not `Element` **of** `nat`.

The type `Any` is the widest type in Mizar, any other is expanded to it. The type `set` in its extension is equal to the type `Any`. The type `DOMAIN` ranges over non-empty sets, that is so-called domains; the type `SUBDOMAIN of D`, over subdomains, that is non-empty subsets; `Real`, over real numbers; `Nat`, over natural numbers.

Now it will be said in general terms what is a type in Mizar.

(1). AN EXPRESSION IN THE FORM:

symbol–of–mode **of** *list–of–terms*

is a type.

Examples:

`Subset` **of** $\mathfrak{G}$, `Subset` **of the** `carrier` **of** $\mathfrak{G}$, `Point` **of** $\mathfrak{G}$, `Subset-Family` **of** $\mathfrak{G}$, `Relation` **of** `X` , `Relation` **of** `X,Y`

If the list of terms is a zero list, then only the symbol of mode will be a type. For instance:

`Any`, `Real`, `TopSpace`, `FinSequence`, `Ordinal`, `Set-Family`,
`Relation`, `Function`.

(2). A SYMBOL OF A STRUCTURE, e.g.:
`TopStruct`, `LattStr`, `IncStruct`,

is a type.

(3). A TYPE IN BRACKETS IS A TYPE, TOO.

In the following examples the types in question are in brackets.

reserve `P` **for** `(Subset` **of** $\mathfrak{G}$`),` `P` **for** `(Point` **of** $\mathfrak{G}$`),` `x` **for** `Any;`
reserve $\mathfrak{J}$ **for** `(Subset_Family` **of** $\mathfrak{G}$`),` `r` **for** `Real;`
reserve $\mathfrak{J}$ **for** `(Subset_Family` **of** $\mathfrak{G}$`),` `P` **for** `Subset` **of** $\mathfrak{G}$`;`
let $\mathfrak{H}$ **be** `(SubSpace` **of** $\mathfrak{G}$`),` `P,Q` **be** `(Subset` **of** $\mathfrak{G}$`),` `x,y` **be** `Any;`

The omission of the brackets in the examples given above does not result in an error. But, for instance, the inscription:

reserve `R` **for** `Relation` **of** `X,` `x` **for** `Any;`

is incorrect. The error consists in the fact that the inscription `Relation` **of** `X` is not in brackets. One might pose the question why the type `Relation` **of** `X` must be in brackets.

Now the mode

`Relation` **of** *list-of-terms*

is defined for the lists which contain zero terms (`Relation`), for lists which contain one term `Relation` **of** `X` and for lists which contain two terms `Relation` **of** `X,Y` . Here are the corresponding definitions:

```
definition
   mode Relation -> set means x ∈ it implies ex y, z st x = [y,z];
end;
```

(`x,y,z` *have the type* `Any`)

```
definition
     let X,Y;
     mode Relation of X,Y -> Relation means it ⊆ [:X,Y:];
end;
```

(`X,Y` *have the type* `set`)

```
definition
     let X;
     mode Relation of X is Relation of X,X;
end;
```

If the inscription *Relation* **of** *list-of-terms* is not brackets, then the *list–of–terms* consists of the maximal number of terms, that is two. Hence the inscription:

reserve `R` **for** `Relation` **of** `X,` `x` **for** `Any;`

is interpreted in the same way as the inscription:

reserve `(R` **for** `Relation` **of** `X,` `x)` **for** `Any;`

but the latter expression is ill–formed because its syntactic structure is incorrect: the last **for** is not preceded by the list of terms.

On the other hand, the modes:

`Subset` **of** *list-of-terms*
`SubSpace` **of** *list-of-terms*
`Subset_Family` **of** *list-of-terms*

are defined only for lists which include one and only one term, which is shown by their definitions:

```
definition
   let 𝔊;
   mode Subset of 𝔊 is set of Point of 𝔊 ;
end;

definition
  let 𝔊;
  mode SubSpace of 𝔊 -> TopSpace means
   Ω(it) c= Ω(𝔊) &
   for P being Subset of it holds P ∈ the topology of it iff
   ex Q being Subset of 𝔊 st Q ∈ the topology of 𝔊 & P = Q ∩ Ω(it);
end;

definition
   let 𝔊;
   mode Subset_Family of 𝔊 is Subset_Family of the carrier of 𝔊 ;
end;
```

Hence in the case of these mode brackets are superfluous.

(4). AN EXPRESSION IN THE FORM:

set **of** *type*

is a type.

Types of those kind are often used in definitions of modes, e.g.,:

```
definition
    let 𝔊;
    mode Subset of 𝔊 is set of Point of 𝔊 ;
end;
```

Types in Mizar have the structure of trees. `X` expands to the type `set`, but neither to `DOMAIN` nor to `SUBDOMAIN` **of** ⌈`DOMAIN`⌉,

`D` expands to `DOMAIN`,

D_1 expands to `DOMAIN`,

`S` expands to `SUBDOMAIN` **of** `D`.

`Real` is adopted as an abbreviation for the type `Element` **of** `REAL` (`Real` is `DOMAIN`), while `Nat` is adopted as an abbreviation for the type `Element` **of** `NAT` (`NAT` is `SUBDOMAIN` of `Real`).

Since the type `Element` **of** `S` expands to the type `Element` **of** `D`, the type `Nat` expands to the type `Real`.

III.3. Reservation of variables

It has been said earlier that in a *Mizar article* it is not allowed to use identifiers of variables for which their type is not given. The type of a given identifier can, for instance, be given in the place of its occurrence, as in the examples:

- **for** `A` **being** `Subset` **of** 𝔊 **holds** `A` ⊆ `Cl A`;
- **let** `x` **be** `Any`, `A` **be** `set`;
- **reconsider** `x` **as** `Real`;

The words **being** and **be** may be used alternately, which is to say that is indifferent from the point of view of Mizar. But in order to be in agreement with the grammar of English certain conventions pertaining to the use of those words have been adopted. For instance, in the Mizar construction **let** ... **be** is used, while **being** is used in quantified formulas.

In order to avoid indicating the type of a given identifier whenever a variable with such an identifier is introduced it is possible to fix that type globally by means of the Mizar construction called the reservation of variables. The reservation of variables has the following form:

reserve *list–of–identifiers* **for** *type* ;

The examples given below show the application of the said construction:

i. If the identifier $\mathfrak{G}$ is to be a variable ranging over topological spaces, then the identifier $\mathfrak{G}$ is to be reserved for the type `TopSpace`. Such a reservation has the form:

reserve $\mathfrak{G}$ **for** `TopSpace;`

If a such a reservation is not made and the identifier $\mathfrak{G}$ is used in the indicated meaning, then the type of that identifier must be given whenever a variable with such an identifier is introduced.

ii. If we want the identifiers `P`, `Q` range over subsets of a topological space $\mathfrak{G}$, then the reservation should be as follows:

reserve `P, Q` **for** `Subset` **of** $\mathfrak{G}$`;`

If we want to reserve identifiers of variables which have various types, then the expression:

list–of–identifiers **for** *type*

in the reservation of variables should be repeated the corresponding number of times and the expression in question must be separated from one another by commas.

This will be illustrated by the following reservations:

reserve `P, Q` **for** `Subset` **of** $\mathfrak{G}$`, x` **for** `Any, p` **for** `Point` **of** $\mathfrak{G}$`;`

The reservation of the identifiers `P`, `Q`, `x`, `p` for the corresponding types may also be as follows:

reserve `P, Q` **for** `Subset` **of** $\mathfrak{G}$`;`
reserve `x` **for** `Any;`
reserve `p` **for** `Point` **of** $\mathfrak{G}$`;`

which is to say that for the various quantifiers one may apply separately the construction *reservation of variables*. That, however, is not the best solution in view of the unnecessary expansion of the text.

Other reservations given by way of example:

- reservation of the identifier `X` for a variable standing for a set:

  reserve `X` **for** `set;`

- reservation of the identifier `x` for a variable standing for a real number:

  reserve `x` **for** `Real;`

- reservation of the identifiers `Z` and `Y` for variables standing for subsets of the set `X`, where the identifier `X` has been earlier reserved or fixed for the type `set`:

  reserve `Z, Y` **for** `Subset` **of** `X;`

- reservation of the identifier `x` for an element of the set of real numbers:

  reserve `x` **for** `Element` **of** `Real;`

Other examples of the construction now under consideration will be given in the chapter dedicated to proofs of theorems.

It must be borne in mind that the reservation of a given identifier for a definite type is made prior to the first occurrence of that identifier. Should we made the reservation later, that is after several occurrences of that identifier, where a variable having that identifier is introduced, the type of that identifier should be given.

The reservation of variables must be made in the text proper. Often the reservation is made at the beginning of the text proper, that is immediately after the word **begin**.

For one and the same identifier the construction *reservation of variables* may be applied several times, according to the need of the author of the article. If, for instance, we first make the reservation:

reserve `x` **for** `Any;`

and in the later part of the Mizar article we change the type of the identifier `x` into `Real` in accordance with the reservation:

reserve `x` **for** `Real;`

then in all occurrence of the identifier `x` between these reservations where the type of the identifier `x` is not given, it will have the type assigned to it in the first reservation, that is `Any`, and in all the occurrences of the identifier `x` after the second reservation, where its type is not given, the identifier `x` will have the type `Real`.

We now proceed to discuss Mizar formulas.

In Mizar there is the following classification of formulas:

- atomic formulas,
- formulas formed of atomic formulas by sentential connectives,
- quantified formulas.

III.4. Atomic formulas

There are several kinds of atomic formulas.

(1). A PREDICATIVE FORMULA,

that is an expression in the form:

list–of–terms *symbol–of–predicate* *list–of–terms*

is an atomic formula.

Here are some symbols of predicates used in articles pertaining to topological space: `is_open`, `is_closed`, `is_open_closed`, `is_dense`, `is_boundary`, `is_nowheredense`, $\subseteq$, `are_separated`.

The number of arguments (both left-side and right-side ones) in the case of each of those predicates is shown in the following table:

symbol of predicate	atomic formula	number of left side arguments	number of right side arguments.
`is_open`	`P is_open`	1	0
`is_closed`	`Q is_closed`	1	0
`is_open_closed`	`Q is_open_closed`	1	0
`is_dense`	`A is_dense`	1	0
`is_boundary`	`B is_boundary`	1	0
`is_nowheredense`	`P is_nowheredense`	1	0
$\subseteq$	`P` $\subseteq$ `Q`	1	1
`are_separated`	`A,B are_separated`	2	0
$\models$	`D,f` $\models$ `H`	2	1

The identifier `D` denotes a family of sets, `f` denotes the valuation of variables by elements of that family, `H` denotes any formula of the language of the ZF set theory. The last formula in the table indicates that in the family `D` the formula `H` is satisfied by the valuation `f`.

Examples of atomic formulas occurred in the table above. Here are other examples:

```
Cl P \ Cl Q ⊆ Cl (P \ Q)
Int P is_open
skl p is_connected
B is_a_component_of 𝔊
𝔉 is_a_cover_of 𝔊
p,q are_joined
p ∈ skl p
```

In particular, because the sign = is the symbol of the predicate, the expression in the form:

term = *term*

called an *equality formula* is an atomic formula.

Examples of *equality formulas:*

```
Fr(Fr(Fr P )) = Fr(Fr P)
Cl(Int P) = Cl(Int(Cl(Int P)))
P^c = Ω𝔊 \ P
P^c = (P qua Subset of the carrier of 𝔊)^c
P \ Q = P ∩ Q^c
Cl (Cl P) = Cl P
```

Likewise, an expression in the form:

term <> *term*

where the sign <> is the symbol of predicate, is an atomic formula.
Examples: `P ∩ Cl Q <> ∅`, `Int (Cl Q) ∩ Int (Cl P) <> ∅`.

(2). An expression in the form:

identifier–of–predicate [*list–of–terms*]

is an atomic formula.

Terms of this kind occur only in the case of private predicates.The identifier of such a predicate is not listed in any vocabulary.

(3). An expression in the form:

term **is** *typ*

is an atomic formula.

A formula in such a form is called *qualifying formula.*

Examples of qualifying formulas:

x **is** Point **of** 𝔊
x **is** set

Here is a theorem NAT_1:1 :

x **is** Nat **implies** x + 1 **is** Nat

where the identifier x has the type Real.
The antecedent and the consequent of the above implication are atomic formulas of the kind under consideration.

III.5. Formulas formed of atomic formulas by propositional connectives

In Mizar the following symbols are used to denote sentential connectives:

not , **&** , **or** , **implies** , **iff** , **contradiction**

which denote respectively:

negation, conjunction, disjunction, implication, equivalence, contradiction.

contradiction is a sentential connective of zero arguments.

Remark: **contradiction** *is treated in Mizar as a formula in the same way as* **thesis** *is.*

Now **not** has the greatest binding force, followed in that respect by &, next by **or**, next by **implies** and **iff** in the same degree. But the binding force of **implies** and **iff** is greater than that of quantifiers.

Since in Mizar the binding force of **implies** and **iff** is the same their simultaneous occurrence in a formula requires the use of the brackets (and) in order to indicate the arguments of the connectives **implies** and **iff** .

Let Φ_1, Φ_2, Φ_3 be atomic formulas.
The formula

$$\Phi_1 \textbf{ implies } \Phi_2 \textbf{ iff } \Phi_3$$

is ill-formed in view of the fact that it is not known which arguments the connectives **implies** and **iff** have. Moreover, the brackets (and) perform in Mizar a role similar to that in arithmetic, which is to say that they indicate the order of the performance of operations.
Examples of formulas formed by sentential connectives:

P is_closed & Q is_closed **implies** Cl(P ∩ Q) = Cl P ∩ Cl Q ,
P is_open **iff** Fr P = Cl P \ P ,
p ∈ P^c **iff not** p ∈ P ,
P ∩ ∅𝔊 = ∅ & ∅𝔊 ∩ P = ∅,
(A is_closed & B is_closed) **or** (A is_open & B is_open)
implies A \ B, B \ A are_separated ,
x ∈ P **implies** x **is** Point **of** 𝔊,
A is_connected & A ⊆ B ∪ C & B,C are_separated
implies A ⊆ B **or** A ⊆ C .

III.6. Quantified formulas

Before we proceed to discuss quantified formulas reference will be made to qualified variables.

In Mizar articles there are often inscriptions which are called *list of qualified variables.* Here are some examples of such lists:

x ,
A,K,n ,
P,Q **being** Subset **of** 𝔊 ,
𝔊 **being** TopSpace, 𝔥 **being** SubSpace **of** 𝔊 ,
𝔉 **being** Subset-Family **of** 𝔊, p **being** Point **of** 𝔊, x, y .

Generally speaking, a list of qualified variables consists of expressions in one of the three forms specified below:

••• *variables–qualified–implicitly*

This name denotes a list of identifiers of variables, that is a finite sequence of identifiers of variables, separated from one another by commas. Examples:

c   x, y, p   A, B

In such cases, as can be seen, the types of the identifiers are not identicated. This means that they are drawn from the list of identifiers which occurs in the reservation of variables.

••• *variables–qualified–explicitly*

At first we explain what a

segment–of–qualified–variables

is. It is an inscription in the form

list–of–identifiers–of–variable **being** (or **be**) *qualification*

Qualification is the type of the identifiers of variables occurring in a segment of qualified variables.

Of course, the type indicated in such an expression is the same for all the quantifiers which occur in it.

Examples of segments of qualified variables:

`a` **being** `Any`
`a,b,c,d` **be** `Any`
`m,n` **be** `Point` **of** $\mathfrak{G}$
`P, Q` **being** `Subset` **of** $\mathfrak{G}$

variables–qualified–explicitly are a finite sequence of segments of qualified variables separated from one another by commas. In the simplest case it is only one segment.

Examples of variables qualified explicitly:

`x1, y1` **be** `Any`
(it is a single segment of qualified variables)

`x` **being** `Real`, `X` **being** `set`
(in this case there are two segments of qualified variables)

$\mathfrak{G}$ **being** `TopSpace`, $\mathfrak{H}$ **being** `SubSpace` **of** $\mathfrak{G}$, `p` **being** `Point` **of** $\mathfrak{G}$
(in this case there are three segments of qualified variables).

••• *variables–qualified explicitly* , *variables–qualified–implicitly*
Examples:

`P, Q` **being** `(Subset` **of** $\mathfrak{G}$`), x, p`

variables qualified explicitly (one segment of qualified variables) — `P, Q` **being** `(Subset` **of** $\mathfrak{G}$`)`
variables qualified implicitly — `x, p`

`P` **being** `(Subset` **of** $\mathfrak{G}$`), p` **being** `(Point` **of** $\mathfrak{G}$`), x, y, z`

variables qualified explicitly (two segments of qualified variables) — `P` **being** `(Subset` **of** $\mathfrak{G}$`), p` **being** `(Point` **of** $\mathfrak{G}$`)`
variables qualified implicitly — `x, y, z`

Obviously, the identifiers of those variables for which types are not given in the examples above must be drawn from the list of quantifiers to be found in the reservation of variables.

Remark: A change of order (variables qualified implicitly preceding those qualified explicitly) is impossible because all variables would become variables qualified explicitly.

In order to explain the above warning we shall consider, by way of example, the following formula:

(•) **for** `A` **being** `(Subset` **of** $\mathfrak{G}$`), x` **st** `x` $\in$ `A` **holds** `x is_Point` **of** $\mathfrak{G}$

In that formula the variable `A` is qualified explicitly while the variable `x` is qualified implicitly. The type of the identified `x` must be given in the reservation. In this case it should be the type `Any`. Should we change the order of the occurrence of the variables `A` and `x` in the formula under consideration, which is to say, should we first give the variable qualified implicitly and next the variable qualified explicitly, we would obtain the formula:

for `x, A` **being** `Subset` **of** $\mathfrak{G}$ **st** `x` $\in$ `A` **holds** `x is_Point` **of** $\mathfrak{G}$

But in that formula the variables `x` and `A` have become variables qualified explicitly; in the process the type of the identifier of `x` has been changed into `Subset` **of** $\mathfrak{G}$ whereas it should be `Any`.

In some case the exchange of variables may be carried out. This will be illustrated by the example of the formula marked (•) above. In its case the formula

for `A` **being** `(Subset` **of** $\mathfrak{G}$`), x` **st** `x` $\in$ `A` **holds** `x is_Point` **of** $\mathfrak{G}$

may be replaced by a formula which has the same meaning in Mizar. Here is that formula:

for `A` **being** `Subset` **of** $\mathfrak{G}$ **for** `x` **st** `x` $\in$ `A` **holds** `x is_Point` **of** $\mathfrak{G}$

Now we can pass to quantified formulas.

A quantified formula (also called a universal sentence) is a formula in which the quantifier occurs openly and is the main sentence-forming functor.

In Mizar the following symbolism was adopted for quantifiers:

for ... **holds** ... – for the universal quantifier,
for ... **st** ... **holds** – for the purified universal quantifier (i.e., universal quantifier with a limited scope),
ex ... **st** ... – for the existential quantifier.

In view of the various forms of the list of qualified variables, a universal sentence, this is a formula in which a universal quantifier occurs, may the following forms:

(A). **for** *identifiers–of–variables* **holds** *formula*

Examples of quantified formulas:

for `P` **holds** `P` $\subseteq$ `Cl P`

(For every subset P *of a topological space* $\mathfrak{G}$ P $\subseteq$ Cl P *holds*
or else
Every subset of a topological space $\mathfrak{G}$ *is included in its closure).*

for `P, Q` **holds** `Cl (P` $\cup$ `Q) = Cl P` $\cup$ `Cl Q`

(For every two subsets P, Q *of a topological space* $\mathfrak{G}$ Cl(P $\cup$ Q) = Cl P $\cup$ Cl Q *holds).*

It can be seen that in each case above the types of the identifiers of the variables are not given openly. When using such formulas in a Mizar article one should bear it in mind that one should previously reserve the identifiers of those variables for the reservation of variables.

But one may also abstain from making earlier the reservation of the variables which occur in a quantified formula. In such a case the types of the identifiers of the variables must be given when the formula is being written. In such a case the form of a quantified formula is as follows:

(B). **for** *segment–of–qualified–variables* **holds** *formula*

A quantified formula has such a form, among other things, if the identifiers of the variables for which no reservation has been made have one and the same type. Otherwise the expression standing between **for** and **holds** must be repeated the corresponding number of times and separated by commas.

Let us consider one case more. Now it may be so that the identifiers of the variables occurring in a quantified formula have being earlier reserved for the corresponding types but when writing the formula we want to apply the same qualifiers of variables to other types. Then such a formula will have the form of the expression shown under (B) above. Let that be illustrated by an example.

Let the following reservation be given:

reserve `A` **being** `SubSpace` **of** 𝔊

In this formula we want to use the identifier `A` which denotes a subset of a topological space 𝔊. Hence the new type o the identifier of `A` must be given in the formula as below:

for `A` **being** `Subset` **of** 𝔊 **holds** `A is_closed` **iff** `Cl A = A`

Examples illustrating the structure of quantified formulas:

1) **for** `P` **being** `Subset` **of** 𝔊 **holds** `P` $\subseteq$ `Cl P`

(For every subset P *of a topological space* 𝔊 P $\subseteq$ Cl P *holds),*

2) **for** `P, Q` **being** `Subset` **of** 𝔊 **holds** `Cl (P` $\cup$ `Q) = Cl P` $\cup$ `Cl Q`

(For any subsets P, Q *of a topological space* 𝔊 Cl(P $\cup$ Q) = Cl P $\cup$ Cl Q)

3) **for** `A` **being** `(Subset` **of** 𝔊`),` `x` **being** `Any` **st** `x` $\in$ `A` **holds** `x is_Point` **of** 𝔊

(For any subset A *of a topological space* 𝔊 *and for any objects* x *which is an element of the subset* A *there holds:* x *is a point of the topological space* 𝔊*).*

The theorem given in example 3) may also be recorded thus:

for `A` **being** `(Subset` **of** 𝔊`)` **for** `x` **being** `Any` **st** `x` $\in$ `A` **holds** `x is_point` **of** 𝔊

EXAMPLES:

- **for** `A` **being** `(Subset` **of** 𝔊`),` `p` **being** `Point` **of** 𝔊 **holds** `p` $\in$ `Cl A` **iff** **for** `C` **being** `Subset` **of** 𝔊 **st** `C is_closed` **holds** `(A` $\subseteq$ `C` **implies** `p` $\in$ `C)`

- **for** 𝔥 **being** `(SubSpace` **of** 𝔊`),` `P, Q` **being** `(Subset` **of** 𝔊`),` `P1, Q1` **being** `Subset` **of** 𝔥 **st** `P = P1 & Q = Q1 & P` $\cup$ `Q` $\subseteq$ Ω𝔊 **holds** `P, Q are_separated` **implies** `P1, Q1 are_separated`

- **for** 𝔥 **being** `(SubSpace` **of** 𝔊`),` `P` **being** `(Subset` **of** 𝔊`),` `Q` **being** `Subset` **of** 𝔥 **st** `P` $\neq$ ∅𝔊 `& P = Q` **holds** `A is_connected` **iff** `B is_connected`

Further, a quantified formula may have the form:

(C).

for *variables–qualified–explicitly* , *variables–qualified–implicitly* **holds** *formula*

There may also be quantified formulas with a purified quantifier. Such formulas are in the form:

for *list–of–qualified–variables* **st** *formula* **holds** *formula*

The structure of the formulas with purified quantifiers (quantifiers with a limited scope) will be illustrated by examples but before their presentation we shall specify several modes and predicates which are introduced in articles pertaining to topological spaces.

They are the modes:

`SubSpace` **of** 𝔊 ,
`Subset-Family` **of** 𝔊

and predicates:

`P, Q are_separated` ,
𝔊 `is_connected` ,
`p, q are_joined`

The above formats of modes have been adopted for denoting, respectively, a subspace of a topological space $\mathfrak{G}$ and the family of the subsets of a topological space. The predicates presented above have been discussed earlier.

We can now pass to the examples.

(i) **for** `P, Q` **st** `P` $\subseteq$ `Q` **holds** `Cl P` $\subseteq$ `Cl Q`
(For any subsets P, Q *of a topological space* $\mathfrak{G}$ *such that* P $\subseteq$ Q *there holds* P $\subseteq$ Q*)*

(ii) `P is_boundary` **iff** (**for** `Q` **st** `Q` $\subseteq$ `P & Q is_open` **holds** `Q = ∅`)
*(*P *is a boundary set if and only if for any open set* Q *included in* P *there holds* Q $= \emptyset$*)*

Had the reservation for the identifiers `P` and `Q` not been made the above formula would be as follows:

> **for** `P` **being** `Subset` **of** $\mathfrak{G}$ **holds** `P is_boundary` **iff**
> (**for** `Q` **being** `Subset` **of** $\mathfrak{G}$**st** `Q` $\subseteq$ `P & Q is_open` **holds** `Q = ∅`)

(iii) **for** $\mathfrak{H}$ **being** `(SubSpace` **of** $\mathfrak{G}$`)`, P_1, Q_1 **being** `(Subset` **of** $\mathfrak{G}$`)`, `P, Q` **being Subset of** $\mathfrak{H}$ **st** `P =` P_1 `& Q =` Q_1 **holds** `P, Q are_separated` **implies** P_1, Q_1 `are_separated`

(For any subspace $\mathfrak{H}$ *of a topological space* $\mathfrak{G}$ *and for subsets* P_1, Q_1 *of the topological space* $\mathfrak{G}$ *and subsets* P, Q *of the subspace* $\mathfrak{H}$*, such that* $P = P_1$ & $Q = Q_1$ *there holds: if* P, Q *are separated, then* P_1, Q_1 *are separated, too.)*

(iv) **for** $\mathfrak{H}$ **being** `(SubSpace` **of** $\mathfrak{G}$`)`, `A` **being** `(Subset` **of** $\mathfrak{G}$`)`, `B` **being** `Subset` **of** $\mathfrak{H}$ **st** `A` $\neq \emptyset\mathfrak{G}$ `& A = B` **holds** `A is_connected` **iff** `B is_connected`
(For any subspace $\mathfrak{H}$ *of a topological space* $\mathfrak{G}$ *and a subset* A *of the topological space* $\mathfrak{G}$ *and for a subset* B *of the subspace* $\mathfrak{H}$ *there holds: if* $A \neq \emptyset\mathfrak{G}$ *and* $A = B$*, then* A *is connected if and only if* B *is connected.)*

A formula which contains the existential quantifier may have one of the three forms listed below:

- **ex** *variables–qualified–implicitly* **st** *formula*
- **ex** *variables–qualified–explicitly* **st** *formula*
- **ex** *variables–qualified–implicitly* , *variables–qualified–implicitly* **st** *formula*

The examples given below contain formulas with the existential quantifier:

`x ∈ Int P` **iff** **ex** `Q` **st** `Q is_open & Q` $\subseteq$ `P & x ∈ Q` ,

(**ex** `x` **being** `Point` **of** $\mathfrak{G}$ **st** **for** `y` **being** `Point` **of** $\mathfrak{G}$ **holds** `x, y are_joined`)
iff (**for** `x, y` **being** `Point` **of** $\mathfrak{G}$ **holds** `x, y are_joined`) .

Other examples of formulas with the existential quantifier will be found later in the text.

We shall now give four topological theorems recorded in English first and next recorded in the Mizar notation.

1) P is boundary set if and only if it is contained in this own boundary.
2) For any subsets P, Q of a topological space $\mathfrak{G}$ such that P $\subseteq$ Q
there holds Cl P $\subseteq$ Cl Q.
3) Any subset A of a topological space $\mathfrak{G}$ is closed if and only if Cl A = A.
4) A point p is in the boundary of a set P if and only if for any open set Q such that p $\in$ Q there holds: the intersection of P and Q is non-empty and the intersection of the complement of P and Q is non-empty.

Here are the above theorems recorded in the Mizar notation:
1) `P is_boundary` **iff** `P` $\subseteq$ `Fr P`
2) **for** `P, Q` **being** `Subset` **of** $\mathfrak{G}$ **st** `P` $\subseteq$ `Q` **holds** `Cl P` $\subseteq$ `Cl Q`
3) **for** `A` **being** `Subset` **of** $\mathfrak{G}$ **holds** `A is_closed` **iff** `Cl A = A`
4) `p` $\in$ `Fr P` **iff**
(**for** `Q` **st** `Q is_open & p` $\in$ `Q` **holds** `P` $\cap$ `Q` $\neq \emptyset$ `&` P^c $\cap$ `Q` $\neq \emptyset$)

The examples given so far in most cases pertained to formulas with a single quantifier, whether universal or existential. But in a formula more than one quantifier may occur, which can be seen in the examples given below.

for `A` **being** `(Subset` **of** $\mathfrak{G}$`), p` **being** `Point` **of** $\mathfrak{G}$ **holds** `p` $\in$ `Cl A` **iff** **for** `G` **being** `Subset` **of** $\mathfrak{G}$ **st** `G is_open` **holds** `p` $\in$ `G` **implies** `A` $\cap$ `G` $\neq \emptyset\mathfrak{G}$
(A point p *of a topological space* $\mathfrak{G}$ *is in the closure of a subset* A *of the topological space* $\mathfrak{G}$ *if and only if for any open subset* G *of the topological space* $\mathfrak{G}$ *which contains the point* p *the intersection of* G *and* A *is non-empty),*

`P is_open` **iff** (**for** `x` **holds** `x` $\in$ `P` **iff** **ex** `Q` **st** `Q is_open & Q` $\subseteq$ `P & x` $\in$ `Q`)
*(*P *is an open set if and only if for any* x*,* x $\in$ P *if and only if there is an open set* Q *such that* Q $\subseteq$ P *and* x $\in$ Q*),*

`P is_closed` **implies** (`P is_boundary` **iff** **for** `Q` **st** `Q` $\neq \emptyset$ `& Q is_open` **ex** `G` **st** `G` $\subseteq$ `Q & G` $\neq \emptyset$ `& G is_open & P` $\cap$ `G = ` $\emptyset$)
(If a set P *is closed, then* P *is boundary set if and only if for any* Q *such that* Q $\neq \emptyset$ *and* Q *is open there is a set* G *such that* G $\subseteq$ C *and* G $\neq \emptyset$ *and* G *is open and* P $\cap$ G = $\emptyset$*),*

for $\mathfrak{J}$ **being** `Subset-Family` **of** $\mathfrak{G}$ **st** $\mathfrak{J} \neq \emptyset$ `&` **for** `A` **being** `Subset` **of** $\mathfrak{G}$ **st** `A` $\in \mathfrak{J}$ **holds** `A is_closed` **holds** `meet` $\mathfrak{J}$ `is_closed`
(The intersection of any $\mathfrak{J}$ *which is a non-empty family of closed subsets of a topological space* $\mathfrak{G}$ *is a closed set),*

for $\mathfrak{J}$ **being** `Subset-Family` **of** $\mathfrak{G}$ **st** (**for** `A` **being** `Subset` **of** $\mathfrak{G}$ **st** `A` $\in \mathfrak{J}$ **holds** `A is_connected`) `&` (**ex** `A` **being** `Subset` **of** $\mathfrak{G}$ **st** `A` $\neq \emptyset(\mathfrak{G})$ `& A` $\in \mathfrak{J}$ `&` (**for** `B` **being** `Subset` **of** $\mathfrak{G}$ **st** `B` $\in \mathfrak{J}$ `& B` $\neq$ `A` **holds not** `A, B are_separated`)) **holds** `union` $\mathfrak{J}$ `is_connected`
(Let $\mathfrak{J}$ *be any family of connected subsets of a topological space* $\mathfrak{G}$ *one of which is non--empty and not separated from any other element of that family. Then the union of elements of that family is a connected set).*

`meet` and `union` are symbols of functors of one arguments each (the recording of the last formula shows that the right-side argument is the only one) which denote, respectively, the intersection and the union of the family of the subsets of a topological space.

The examples given so far have been drawn from the articles `PRE_TOPC`, `TOPS_1` and `CONNSP_1`, which self–evidently pertain to problems connected with topological spaces.

Let us revert once more to general sentences with a purified quantifier. Such a sentence can be recorded, of course, in a different manner without an change in its meaning. We mean the elimination of the limited range of the quantifier in a general sentence and the replacement of the condition by implication.

Let Φ_1, Φ_2 be any formulas. A general sentence (a sentence in which a universal quantifier occurs):

for *list–of–qualified–variables* **st** Φ_1 **holds** Φ_2

is equivalent to the sentence:

for *list–of–qualified–variables* **holds** Φ_1 **implies** Φ_2

Now the formula

Φ_1 **implies** Φ_2

is not bracketed because the binding force of **implies** (like that of **iff**) is greater than that of quantifiers.

EXAMPLES:

The formula

for `P, Q` **st** `P` $\subseteq$ `Q` **holds** `Cl P` $\subseteq$ `Cl Q`

has for Mizar the same meaning as the formula

for `P, Q` **holds** `P` $\subseteq$ `Q` **implies** `Cl P` $\subseteq$ `Cl Q`

because both formulas have one and the same semantic correlate (see III.7).

Likewise formula:

`P is_boundary` **iff** `(`**for** `Q` **st** `Q` $\subseteq$ `P & Q is_open` **holds** `Q =` $\emptyset$`)`

has for Mizar the same meaning as the formula

`P is_boundary` **iff** `(`**for** `Q` **holds** `Q` $\subseteq$ `P & Q is_open` **implies** `Q =` $\emptyset$`).`

The theorems in which the universal quantifier occurs openly can be recorded as non-quantified formulas. For instance, the theorem:

for `P` **being** `Subset` **of** $\mathfrak{G}$ **holds** `Int P = P` $\setminus$ `Fr P`

can be recorded thus:

`Int P = P` $\setminus$ `Fr P`

because both sentences have the same meaning for Mizar (see semantic correlates).

Likewise the sentences:

for $\mathfrak{G}$`, P` **holds** `Int P = (Cl (P`^{c}`))`^{c} ,

for `P` **holds** `Int P = (Cl (P`^{c}`))`^{c} ,

`Int P = (Cl (P`^{c}`))`^{c}

will all be read in the same way by the system (if $\mathfrak{G}$ and `P` have not been fixed earlier).

The formula:

`Int P = (Cl (P`^{c}`))`^{c}

will be read by the system as the formula:

for $\mathfrak{G}$`, P` **holds** `Int P = (Cl (P`^{c}`))`^{c}

The various forms in which formulas are recorded have significance only for the author of a given article. Some of them may be more legible, but the processor of PC Mizar transforms the formulas it reads and brings them to a certain fixed for (see III.7).

Here are other examples illustrating the different forms of recordings of Mizar formulas:

1) **for** `P, Q` **st** `P` $\subseteq$ `Q` **holds** `Cl P` $\subseteq$ `Cl Q`

can be recorded thus:

P $\subseteq$ Q **implies** Cl P $\subseteq$ Cl Q

2) **for** `P, Q` **st** `P is_dense & Q is_dense & Q is_open` **holds** `P` $\cap$ `Q is_dense`

(if `P, Q` have not been fixed earlier)

can be recorded thus:

P is_dense & Q is_dense & Q is_open **implies** P ∩ Q is_dense

The antecedent of the implication is not bracketed because conjunction has a greater binding force than implication has.

3) **for** P **st** P is_open **holds** Cl(Int(Cl P)) = Cl P
can be recorded thus:

P is_open **implies** Cl(Int(Cl P)) = Cl P

In all the examples recorded in the new version the quantifier is understood.

The formulas given in the above examples, if not recorded with the use of the quantifier, will be processed by the system into quantified ones (see III.7). The identifiers of variables will follow the word **for** in the formula P ⊆ Q **implies** Cl P ⊆ Cl Q the identifier of P will come first, followed by the identifier of Q (if the space 𝔊 has not been fixed earlier, then the identifier of 𝔊 will additionally processed in to such a quantified formula in which the word **for** is first followed by the identifier of P (or the identifiers of 𝔊 and P), and next by the identifier of Q as under (1) above. But sometimes it is so that the required sequence of the identifiers differs from that arranged automatically. In such a case a given formula should be written in the desired quantified quantified version.

The word **holds** before the word **ex** or before the word **for** may be omitted. Hence the formula:

for ...... **holds ex** ......

may be recorded as below, by replacing the expression **holds ex** by the word **ex** :

for ...... **ex** ......

Examples:
The formulas:

1) **for** A **being** Subset **of** 𝔊 **st** A ≠ ∅𝔊 **holds ex** x **being** Point
of 𝔊 **st** x ∈ A
2) P is_closed **implies** (P is_boundary **iff for** Q **st** Q ≠ ∅ & Q is_open
holds ex G **st** G ⊆ Q & G ≠ ∅ & G is_open & P ∩ G = ∅)

may be recorded, in accordance with what has been said, in the following manner:

1) **for** A **being** Subset **of** 𝔊 **st** A ≠ ∅𝔊 **ex** x **being** Point **of** 𝔊 **st** x ∈ A
2) P is_closed **implies** (P is_boundary **iff for** Q **st** Q ≠ ∅ & Q is_open
ex G **st** G ⊆ Q & G ≠ ∅ & G is_open & P ∩ G = ∅)

Likewise a formula in the form:

for ...... **holds for** ......

may be recorded:

for ...... **for** ......

where the expression **holds for** has been replaced by **for** .

For instance, the formula:

for 𝔋 **being** SubSpace **of** 𝔊 **holds for** A **being** Subset **of** 𝔋
holds A **is** Subset **of** 𝔊

may be replaced by the formula:

for 𝔋 **being** SubSpace **of** 𝔊 **for** A **being** Subset **of** 𝔋
holds A **is** Subset **of** 𝔊

The examples given so far show that theorems may be recorded in several ways. The choice of the form of the recording depends on the author of the article. It is recommended to use such a recording of the content of a given theorem which would be the most legible and practical. For instance, the use in a general sentence of a purified quantifier (trough the use of the word **st**) sometimes increases its legibility. The same applies to the case in which we indicate the types of identifiers of variables when writing a formula. The reservation of variables is made at the beginning of a given article or later in the text. If the article is long, then when reading a theorem (contained in it) in which the types of the variables are not indicated we have to look for the reservations in the text, and that means an unnecessary loss of time.

III.7. Semantic correlates

The PC Mizar processor transforms the formulas (terms, types) it reads into certain standard forms. The form of a formula obtained by such a transformation is called the semantic correlate (semantic form) of that formula. To make the transformation of formulas (terms, types) possible a certain relation of equivalence has been defined on formulas. It states that two formulas between which that relation holds will be transformed in the same way. The classes of abstraction of that relation of equivalence are called semantic correlates. If two formulas are in one and the same class of abstraction then this means that they have the same semantic correlate. From among the formulas which form a given class of abstraction one can choose formula which is the standard representation of that class of abstraction. Such a formula is formed by the signs of negation (**not**), conjunction (`&`), **not contradiction**, i.e., `VERUM`, and base sentences, i.e., atomic formulas and general sentences.

Moreover conjuction and negation satisfy the conditions:

1. Conjunction is associative, which is to say that for any formulas α_1, α_2, α_3 the formulas

$$(\alpha_1 \ \& \ \alpha_2) \ \& \ \alpha_3 \qquad \text{and} \qquad \alpha_1 \ \& \ (\alpha_2 \ \& \ \alpha_3)$$

are in the same class abstraction, that is they have one and the same semantic correlate.

2. Negation is an involution, so that for any formula α the formulas

$$\textbf{not not } \alpha \qquad \text{and} \qquad \alpha$$

have one and the same semantic correlate.

3. If a free variable, that is such which is not openly bound by a quantifier, occurs in a given formula, then the universal quantifier is automatically prefixed to that formula.

For instance, if we write the formula α(`x`), in which `x` is a free variable (i.e., not bound by a quantifier), then that formula will be read by the system as the formula **for** `x` **holds** α(`x`). Hence the formulas

$$\alpha(\mathtt{x}) \qquad \text{and} \qquad \textbf{for } \mathtt{x} \textbf{ holds } \alpha(\mathtt{x})$$

have one and the same semantic correlate.

The formula **contradiction** has **not** VERUM as its semantic form.

The semantic correlates of predicative formulas except for the formulas in the form

term = *term*

is the same original (initial) form.

The semantic correlates of the predicative formula in the form

term <> *term*

is the formula

not *term* = *term*

The formula in the form

term <> *term*

is the antonym of the formula in the form

term = *term*

Moreover, for the formula `x` $\leq$ `y` (where `x, y` have the type `Element` **of** `REAL`) there are two antonyms:

`x` $>$ `y` and `y` $<$ `x`

which are synonyms, and the synonym: `y` $\geq$ `x`.

The knowledge of semantic correlates can be used in the construction of skeletons of proofs, because the form of the semantic correlate of a given formula determines the skeleton of the proof of that formula.

If `P`, `Q`, `A`, `B`, `C`, $\mathfrak{G}$ are not constants but earlier reserved identifiers of variables, then the formulas

a) `P` $\subseteq$ `Cl P` and **for** `P` **holds** `P` $\subseteq$ `Cl P`

b) `Int Q is_open` and **for** `Q` **holds** `Int Q is_open`

c) $\Omega(\mathfrak{G}) \setminus$ `A = B` $\cup$ `C & B,C are_separated & A is_closed` **implies** `A` $\cup$ `B is_closed & A` $\cup$ `C is_closed`

and

for $\mathfrak{G}$`, A, B, C` **holds** $\Omega(\mathfrak{G}) \setminus$ `A = B` $\cup$ `C & B,C are_separated & A is_closed` **implies** `A` $\cup$ `B is_closed & A` $\cup$ `C is_closed`

have the same semantic correlates correspondingly in the examples a), b), and c).
In the formulas α and β have one and the same semantic correlate, then α may be replaced by β and conversely. This is advantageous, because if we want to prove α it is sometimes more convenient to prove β.
Here are several pairs of formulas:

α & **not contradiction**	and	α
α **implies contradiction**	and	**not** α
not contradiction implies α	and	α
for `x` **for** `y` **holds** α	and	**for** `x, y` **holds** α
ex `x` **st ex** `y` **st** α	and	**ex** `x, y` **st** α
for `x` **st** α **holds** β	and	**for** `x` **holds** α **implies** β
α & β **implies** γ	and	α **implies** (β **implies** γ)
not not α	and	α
α **or** β	and	**not** α **implies** β
not ex `x` **st** α	and	**for** `x` **holds not** α
α **iff** β	and	(α **implies** β) & (β **implies** α)

Formulas in each pair have the same meaning for Mizar. They are thus formulas which have the same semantic correlate.

Remark:

The sentences α & β *and* β & α *have different semantic forms. The same applies to the sentences*

α **or** β *and* β **or** α .

In the above examples the formulas α, β, γ should, in order to secure the correct construction of sentences and the subsumption of those sentences under the given sentence schemata, be bracketed whenever necessary. Should, for instance, γ be an implication or equivalence, then the formulas in which it would occur should be written thus:

α & β **implies** γ and α **implies** (β **implies** γ) .

The same applies to α and β.

IV. PROVING SENTENCES IN MIZAR

IV.1. Justifications

Before proceeding of that theorem in the Mizar notation and then proceed to justify it.

There are several possibilities of justifying theorems, but at this point we shall be concerned with only one them, namely straightforward justification is a justification in which one gives the reference (list of labels indicating the sentences which are the premisses of the theorem being justified). Straightforward justification can be classed into:

a) simple justification,
b) justification by schema.

Direct justification has been following form:

(∗) *sentence–justified* **by** *list–of–references* ;

The list of references is a finite sequences of references separated from one another by commas.

References have been discussed earlier. Note only that they are classed into library references (which to theorems to be found in articles) and local references (which through labels enable one to use sentences justified earlier and to be found in a given article).

We shall give below several sentences justified directly:

(1) `M ∪ ∅ = M` **by** `BOOLE:60;`

`BOOLE:60` is a library reference. It denotes the theorem No.60 to be found in the file `BOOLE.abs`.

(2) `k + l = l + k` **by** `NAT_1:3;`

(3) `k ≤ 0 & 0 ≤ 1` **implies** `k ≤ 1` **by** `NAT_1:13;`

(see example No.2 in the file `art.lst`).

Remark:

The justification by **by** *should include labels of sentences which have occurred earlier (in an earlier part of the text or in an earlier article) and are accessible in the place of reference (which is to say that they are labels which occurred at an earlier closed level of reasoning (1) or point to the current level of reasoning (2)). Hence the justifications in the following example would be incorrect:*

EXAMPLE

```
for M, N being set, x being Any st x ∈ M holds x ∈ M ∪ N
proof
    let M,N be set, x be Any;
    assume A: x ∈ M;
    hence thesis by BOOLE:8;
end;

for x being Any, M being set holds x ∈ M by A;
                                            * (1)
B: now
```

```
let x be Any, M be set;
x ∈ M implies x = x by B;
                              * (2)
end;
```

(see example No. 3 in `art.lst`*).*

In the case of some theorems it is more convenient, before one proceeds to prove them, to prove, earlier (a) auxiliary lemma(s). Then proof of the theorem proper will offer no problems because it will be a straightforward justifications.

For instance, if one wants to prove a theorem which is an equivalence, then one can earlier prove the necessary implications. Such a case is presented in example No. 30 in the annex.

Sometimes it is convenient to justify an auxiliary lemma (or lemmas) in the process of proving a given sentence. It is also worth mentioning such straightforward justification in which the reference list is a zero list. In such a case (∗) has the form:

justified–sentence ;

That special kind of justification pertains only to those sentences which are tautologies of the propositional calculus or simple laws of the functional calculus.

That part of the system which verifies justifications is called CHECKER. Tautologies are self-evident for CHECKER and require no justification.

Straightforward justification with a zero reference will be illustrated by examples.

```
Int P = P implies not (Int P ≠ P & Cl P = P);
P = Q implies (P is_open iff Q is_open);
for k, l holds k = l or k ≠ l;
```

(see example No. 36 in the file `art.lst`).

Justification by schema in the following form:

justified–sentence **from** *symbol–of–schema* (*reference–list*) ;

is another straightforward justification.

The number of references in a reference list may be zero, as in any expression in the form list–... .

If the reference list is a zero list, then justification by schema has the following form:

justified–sentence **from** *symbol–of–schema* ;

Example four in the file `art.lst` illustrates the proof of a theorem in which the schema of induction is used.

In the examples given above we had to do with straightforward justification only. But in most cases a theorem requires a proof, and straightforward justifications – especially a direct one – find application in the reasoning used in the proof (is a certain step in the proof).

If the truth of a theorem cannot be justified directly or by reference to a schema, then a proof must be carried out.

After recording the content of the theorem we write:

```
proof
   ...
end;
```

where the dots will, of course, be replaced by a certain reasoning.

Every reasoning is a sequence of successive transitions must be justified (straightforward justifications or by proof). Exceptions in that respect are those recordings which form the skeleton of the proof (assumption, generalization, exemplification), but these will be discussed in the next section.

A justified step in a proof is called a statement. The steps which combine to from the proof depend on the thesis of the theorem, the way of proving (e.g. direct or indirect proof) and, obviously, the imagination of the person who writes the article.

Let us try to prove (without restoring to the Mizar notation) the following topological proof:

For any subset A, B of a topological space 𝔊 the following holds:

$$\mathrm{Cl}\ (A \cap B) \subseteq \mathrm{Cl}\ A \cap \mathrm{Cl}\ B$$

Proof .

Let us consider any two subsets A, B of topological space 𝔊.

It is know from the properties of sets that $A \cap B \subseteq A$ and $A \cap B \subseteq B$. By availing ourselves of the following property of the closure operation:

If $M \subseteq N$ then $\mathrm{Cl}\ M \subseteq \mathrm{Cl}\ N$ where M, N are subsets of a topological space we may write:

$\mathrm{Cl}\ (A \cap B) \subseteq \mathrm{Cl}\ A$ and $\mathrm{Cl}\ (A \cap B) \subseteq \mathrm{Cl}\ B$

Hence $\mathrm{Cl}\ (A \cap B) \subseteq \mathrm{Cl}\ A \cap \mathrm{Cl}\ B$

quod erat demonstrandum.

Let us now try to record that proof in the Mizar notation. The Mizar article which would carry the proof of the theorem under consideration would have, as is know, to consist of the following elements:

environ
directives of environment
begin
content of theorem
proof
reasoning
end;

The theorem in question, when recorded in Mizar notation, has the following form:

```
for A, B being Subset of 𝔊 holds Cl (A ∩ B) ⊆ Cl A ∩ Cl B
```

In accordance with what has been said in the chapter concerned with formulas the above recording of the content of the theorem is only one of several possible versions.

We shall now proceed to construct the next proper, that is the text which follows the word **begin**. The remaining part of the article will be discussed later. Such a sequence of writing the proof is of a certain importance, especially for a person who starts learning Mizar. Now after the writing of the text proper one can see clearly which directives of the environment must be inserted between the words **environ** and **begin**. But, on the other hand, it must be borne in mind that if the environment is defectively constructed during a considerable time taken by the process of proving, than that will make the proof more difficult because in the *Mizar procedure*, that is the verification of the correctness of the proof in progress, errors related to the defective construction of the environment will be reported.

The construction *reservation of variables* is used only for the identifier of 𝔊, which indicates a certain topological space. The types of the remaining identifiers will be given whenever necessary.

The next proper then assumes the form:

```
reserve 𝔊 for TopSpace;
for A, B being Subset of 𝔊 holds Cl (A ∩ B) ⊆ Cl A ∩ Cl B
    proof
        reasoning
    end;
```

At this point the carrying out the reasoning remains.

In the previous proof we considered any two subsets of a topological space 𝔊. Now we shall proceed analogically.

After the word **proof** we have to write:

let `A, B` **be** `Subset` **of** `𝔊;`

This expression can be translated thus:

Let A, B be any subsets of a topological space 𝔊.

In the expression **let** `A, B` **be** `Subset` **of** `𝔊;` the types of the variables `A, B` had to be specified because they had not been reserved in the reservation of variables.

(The qualification given in a formula has its scope only until the and of that formula.)

We continue to imitate the previous proof and write:

`A ∩ B ⊆ A & A ∩ B ⊆ B;`

This is a certain step in the reasoning. It has been said earlier that every step of the reasoning. It has been said earlier that every step of the reasoning must be justified because otherwise the CHECKER will report error No. 4: `This reference is not accepted by Checker`.

In the case under consideration direct justification will suffice; this is to say we mean a justification which does not require indication of the appropriate references.

Note once more that the files in which the contents of the theorems are in the subdirectory `\ABSTR` (`\ABSTR` is a subdirectory of `\MIZAR`) and have the extension `*.abs`. When inspecting the file `BOOLE.abs` we come across Theorem No.37 (i.e., `BOOLE:37`), which states that for any sets `X,Y` we have:

`X ∩ Y ⊆ X & X ∩ Y ⊆ Y`

when reference is made to this theorem the first step of the reasoning is justified. We obtain the statement:

`Z1: A ∩ B ⊆ A & A ∩ B ⊆ B` **by** `BOOLE:37;`

The next justified step of the reasoning we obtained by the application of Theorem No.49, to be found in the file `PRE_TOPC.abs`. Its content is:

for `A, B` **being** `Subset` **of** `𝔊` **st** `A ⊆ B` **holds** `Cl A ⊆ Cl B`

This theorem is to be applied to this formula

`A ∩ B ⊆ A & A ∩ B ⊆ B`

That is why it was necessary to provide it with a label, which in our case consists of the inscription `Z1`. Note that the identifier of a label must be followed by a colon `:`. By referring to a given label we refer to the sentence which bears that label.

The second step in the reasoning will be as follows:

`Z2: Cl (A ∩ B) ⊆ Cl A & Cl (A ∩ B) ⊆ Cl B` **by** `Z1, PRE_TOPC:49;`

As can be seen, this sentence has been provided with a label because it will have to be used as a premiss in the further part of the proof.

As can be seen, the theorem `PRE_TOPC:49` has been applied twice, but in the justification it has been given only once.

Remark: If in one and the same step of the proof a reference is indicated several times, then it suffices to give it only once after the word **by**.

After availing ourselves of the theorem `BOOLE:39`, which says:

`Z ⊆ X & Z ⊆ Y` **implies** `Z ⊆ X ∩ Y`

(where `X`, `Y`, `Z` are any sets)

we can write down the conclusion:

thus `Cl (A ∩ B) ⊆ Cl A ∩ Cl B` **by** `Z2, BOOLE:39;`

This is the last step in the reasoning of the proof.

The word **thus** precedes the sentence which is the thesis of the proof or its part. In our case it is the thesis of the proof.

Ultimately, the next proper is as follows:

```
reserve 𝔊 for TopSpace;
for A, B being Subset of 𝔊 holds Cl (A ∩ B) ⊆ Cl A ∩ Cl B
    proof
      let A, B be Subset of 𝔊;
      Z1:  A ∩ B ⊆ A & A ∩ B ⊆ B by BOOLE:37;
      Z2:  Cl (A ∩ B) ⊆ Cl A & Cl (A ∩ B) ⊆ Cl B by Z1, PRE_TOPC:49;
      thus Cl (A ∩ B) ⊆ Cl A ∩ Cl B by Z2, BOOLE:39;
    end;
```

It now remains to insert the appropriate directives of the environment between the word **environ** and **begin**. Let us begin with the vocabularies, that is, with the directive **vocabulary** ... ;.

The text proper above there the following symbols occur:

∩ , Cl	symbols of functors,
⊆	symbol of predicate,
TopSpace, Subset	symbols of modes.

The symbol ∩ is to be found in the vocabulary BOOLE. The symbols Cl and TopSpace are introduced in the vocabulary TOPCON, whereas the symbol of the mode Subset and of the predicate ⊆ are in the vocabulary HIDDEN. The vocabulary directives which must be included in our article are:

vocabulary BOOLE;
vocabulary TOPCON;

The directive **signature** α; allows on to use vocabulary symbols in accordance with the format defined in the article α.miz, format - the number of left-side and right-side arguments and also the types of the result and the arguments of the functor or an expansion of a mode.

In our case the symbol ∩ is used as that of the intersection of sets. That intersection may be treated as an intersection of subsets of a topological space which yields also a subset of that space, or else – without any modification in the reasoning – as an ordinary intersection of sets. Hence its use requires the joining of the directive:

signature SUBSET_1;
(*redefinition for subsets*)

or the directive

signature PRE_TOPC;
(*redefinition for subsets of a topological space*)

or the directive

signature BOOLE;
(*definition of intersection of sets*).

The definition of the predicate of inclusion, for which we use the symbol ∩ , is to be found in the article TARSKI. If it is to be used in our case one of the directives of environment must be

signature TARSKI;

In the article PRE_TOPC the definition of the closure of a set, symbolized Cl, and the definition of the mode TopSpace, are introduced, hence it is necessary to join the directive

signature PRE_TOPC;

Since in the proof we availed ourselves of theorems to be found in the articles BOOLE and PRE_TOPC, two more directives must be added to the earlier given directives of environment, namely:

theorems BOOLE;
theorems PRE_TOPC;

The Mizar article containing the proof of the theorem under consideration is to be found in the annex, example 5 in the file `art.lst`.

Remark: In Mizar it is allowed to overridden labels. Hence the marking of several sentences from one and the same level of reasoning is not an error.

If, at a given level of reasoning, in which there are no other levels of reasoning, several sentences are marked by the same label, then the reference to that label means reference to the last sentence marked by it.

By a level of reasoning we mean:

(a) the reasoning contained between the correspondingly paired words **proof** and **end;**,
(b) the reasoning contained between the correspondingly paired words **now** and **end;**,
(c) the reasoning contained between the correspondingly paired **schemes** and **;**.

Reference to labels from an earlier closed level of reasoning is not allowed.

EXAMPLE

Two levels of reasoning will be shown below. The inscriptions α, β, γ, δ denote certain formulas. At the shown levels of reasoning the label A: occurs only in the indicated positions.

```
+-------------------------------------+
|          Level of reasoning         |
+-------------------------------------+
|      . . . .                        |
|      A: α;                          |
|      . . . .                        |
|      +-------------------------+    |
|      |   Level of reasoning    |    |
|      +-------------------------+    |
|      |      . . . .            |    |
|      |      A: β;              |    |
|      |      . . . .            |    |
|      |      γ by A;            |    |
|      |      . . . .            |    |
|      +-------------------------+    |
|      . . . .                        |
|      δ by A;                        |
|      . . . .                        |
+-------------------------------------+
```

In the justification of the sentence γ the reference to the label means reference to the last sentence marked by it, that is to the sentence β. But in the justification of the sentence δ the reference to the label A: means reference to the sentence α. At the place it is impossible to refer to the second sentence marked by the label A:, that is to the sentence β, because that sentence is at the previously closed level of reasoning.

In the proofs of theorems are long it is convenient to use the corresponding recording of the reasoning used in the proof, that is such which shortens that reasoning and makes it clearer and more legible. That can be achieved by the elimination of the labelling of sentences through the use in the proof of the words: **then**, **hence**, and **thesis**.

The prefixing of the sentence β by the word **then** indicates that in the justification of β we avail ourselves of the sentence α which directly precedes β. In such a case α need not be marked by a label.

This way of justification is called *linking*.

We shall use *linking* in the proof carried out earlier. Instead of the sentence labelled `Z1:` and `Z2:` we shall have:

`A ∩ B ⊆ A & A ∩ B ⊆ B` **by** `BOOLE:37;`

then `Cl (A ∩ B) ⊆ Cl A & Cl (A ∩ B) ⊆ Cl B` **by** `PRE_TOPC:49;`

It must be emphasized that *linking* requires that the expression β be a statement justified straightforwardly, and α-sentence. That imposes certain limitation upon the application of linking. For instance, linking cannot be applied directly after **proof** not after a collective assumption, because it is not know which of the partial assumptions is meant. Not can it be applied after the statement of choice, after the statement of a change of type, after exemplification. Linking can be applied directly after a sentence, a statement justified directly a statement of choice, and a diffuse statement.

If the preceding sentence is one of the premisses of the conclusion, then linking may be indicated by the replacement of **thus** by **hence**. The sentence preceding the conclusion may be unlabelled. Figuratively speaking, the recording:

`A:`α`;`
thus β **by** `A,` *other–references*`;`

may be replaced by the recording:

α`;`
hence β **by** *other–references*`;`

When the possible linkings are considered the proof of the theorem discussed earlier will assume the form:

```
proof
    let A, B be Subset of 𝔊;
    A ∩ B ⊆ A & A ∩ B ⊆ B by BOOLE:37;
    then Cl (A ∩ B) ⊆ Cl A & Cl (A ∩ B) ⊆ Cl B by PRE_TOPC:49;
    hence Cl (A ∩ B) ⊆ Cl A ∩ Cl B by BOOLE:39;
end;
```

(see example No. 6 from `art.lst`)

The conclusion (that is the thesis of the proof or its part), which in this case in the expression

`Cl (A ∩ B) ⊆ Cl A ⊆ Cl B`

may be replaced by the word **thesis**, which means *that which is left to be demonstrated.*

The word **thesis** is treated by Mizar as a formula. The formula **thesis** may be used solely within the proof, that is between the words **proof** and **end;** .

EXAMPLES of the use **thesis**

- *in the termination of the proof:*

  ... **hence thesis; end;** or ... **thus thesis; end;**

The example Nos. 7, 10, 28, 29 from the file `art.lst` illustrate the application of the formula **thesis** at the end of the proof.

In the example No. 7 in the first inner proof, that is in the proof of the thesis `P∪ΩT=ΩT`, **thesis** denotes the formula `P∪ΩT=ΩT`, whereas in the second inner proof **thesis** denotes the formula `P∪ΩT=P`.

In the example No.10 the formula **thesis** occurs twice. In the first case it denotes the formula `W is_open & W ⊆ P & x ∈ W`, whereas in the second case, at the end of the proof, it denotes the formula `x ∈ Int P`.

In the example No.28 **thesis** denotes the sentenced being proved in that examples, that is the sentence

`T is_connected` **iff for** `A` **being** `Subset` **of** `T` **st** `A is_open_closed`
holds `A = ∅T` **or** `A = ΩT`

whereas **thesis** in the example No.29 denotes the formula `P is_dense`.

- *at the beginning of an indirect proof:*

  proof assume not thesis; ... ; **thus contradiction**; **end**;

Such a use of **thesis** found application in the examples Nos. 19, 22, 23.

In the example No.19 **thesis** denotes the sentence being proved, that is the formula:

for G **st** G is_open **holds** p $\in$ G **implies** P $\cap$ G $\neq \emptyset$.

In the example No.22 it denotes the formula P $\neq$ Q, and in the example No.23, the formula A $\neq$ $\emptyset$T.

- *in a proof by cases:*

```
proof
      A: now assume α; ... ; hence thesis; end;
         now assume not α; ... ; hence thesis; end;
      hence thesis by A;
end;
```

IV.2. Skeletons of proofs

Every proof, that is the reasoning contained between **proof** and **end** consists of elements which from its skeleton. The skeleton of a proof consist of:

assumption,
generalization,
conclusion,
exemplification.

It is to be noted that the skeleton of a proof is not determined unambiguously. Its structure depends on the form of the thesis to be proved and on the technique of proving (for instance, the *direct* or the *indirect proof*). The skeleton of the proof of a given thesis is based on the structure of the semantic correlate of that thesis. That part of the system which verifies the correctness of the structure of the semantic correlate of that thesis. That part of the system which verifies the correctness of the structure of the skeleton is called REASONER.

When preceding to prove a sentence it is advisable to write at first the correct skeleton of the proof of that sentence (that is to say, disregarding the justifications of sentence). If the skeleton of the proof is written correctly, then only the errors marked by number 4 will be reported (by CHECKER).

IV.2.1. DIRECT PROOFS

We shall now show the likely skeletons of proofs when the thesis is a *conjunction, disjunction, implication, equivalence, a general sentence,* and *an existential sentence.*

1. *CONJUNCTION,* that is an expression in the form α_1 & α_2, where α_1 and α_2 are any formulas, is a thesis.

If this sentence is to be proved *directly*, then the skeleton of the proof may consist of the expression listed below and contained between the words **proof** and **end** :

```
proof
      ........
      thus α1 ;
      ........
      thus α2 ;
      ........
end;
```

The dots indicate that they are to be replaced by the remaining steps of the proof.

Self-evidently, the sentences α_1 and α_2 must be justified by a straightforward justification or by proof.

Every expression which is a component of the skeleton of the proof of a given sentence modifies the thesis of the proof. In the example given above until the occurrence of the expression **thus** α_1 the formula

$$\alpha_1 \ \& \ \alpha_2$$

was the thesis of the proof. But the expression **thus** α_1 modified the thesis of the proof, which after that expression became the formula α_2.

The sequence of the justification of α_1 and α_2 is essential; it must be such as presented above. If we change the order into:

thus α_2 ;
thus α_1 ;

then we obtain the skeleton of a proof of the sentence $\alpha_2 \ \& \ \alpha_1$. But the semantic correlates of the sentence

$$\alpha_1 \ \& \ \alpha_2 \qquad \text{and} \qquad \alpha_2 \ \& \ \alpha_1$$

are different. Hence for Mizar these are two different sentences.

The example `Z8.lst` shows the form of the file when only the skeleton of the proof of the sentence being justified is written down.

If in the proof in that example we change the order of the expressions which for the skeleton of the proof, then additionally the error marked by No.51 – `Invalid conclusion` – will be reported. Such a situation is illustrated by the example `Z9.lst`.

The full proof is shown in the example No. 7 from the file `art.lst`.

In the proof of the thesis in that example there are two inner proofs. In each of them the final conclusion of each proof is marked by the word **thesis**. In the first inner proof it denotes the formula `P∪ΩT = ΩT`, and in the second, the formula `P∩ΩT = T`.

If the thesis is a conjunction of more than two constituents, then the truth of each constituent is to be justified.

For instance, for the thesis

$$\alpha_1 \ \& \ \alpha_2 \ \& \ \alpha_3$$

the skeleton of the proof might be as follows:

proof
....
thus α_1 ;
....
thus α_2 ;
....
thus α_3 ;
....
end;

Remark:

The skeleton of the proof for the thesis $\alpha_1 \ \& \ \alpha_2 \ \& \ \alpha_3$ *is also the skeleton of the proof for the thesis* $\alpha_1 \ \& \ (\alpha_2 \ \& \ \alpha_3)$ *and for the thesis* $(\alpha_1 \ \& \ \alpha_2) \ \& \ \alpha_3)$*, which is to say that all the three formulas have one and the same semantic correlate.*

2. *IMPLICATION* is a thesis.

There are two methods of proving implications, the *direct* and the *indirect*.

If the implication

$$\alpha_1 \ \textbf{implies} \ \alpha_2$$

is to be proved directly, then one has to assume the antecedent of the implication and prove the consequent. Hence the skeleton of the proof of the above sentence will be as follows:

proof

....

assume α_1 ;

....

thus α_2 ; *(conclusion)*

end;

Since the assumption in part of the skeleton of the proof it modifies the thesis. Before the assumption the thesis was the formula

α_1 **implies** α_2

but by assuming the antecedent of the implication we modify the thesis after the assumption the thesis become the formula α_2.

EXAMPLE

The skeleton of the proof of the sentence

```
ΩG = A ∪ B & A is_closed & B is_closed & A ∩ B = ∅G
     implies A, B are_separated
```

may be such:

```
proof
    ......
    assume ΩG = A ∪ B & A is_closed & B is_closed & A ∩ B = ∅G;
    ......
    thus A, B are_separated;
end;
```

(see annex – file `Z10.lst`)

Remark:

The steps which constitute the skeleton of the proof (except for the conclusion) do not require justification. The remaining steps of the proof other that tautologies must be justified. This is explained by the example `Z10.lst` *from the annex, in which the error connected with the justification of the assumption is not reported.*

The expression

assume `ΩG = A ∪ B & A is_closed & B is_closed & A ∩ B = ∅G;`

is a *single assumption*, which is one of the forms of the assumption.

A *single assumption* may take on one of the two forms presented below:

assume *sentence* ;

Such an assumption is used when we refer to it by linking. But sometimes it is not possible to refer to the sentence in the assumption by linking. In such a case that sentence must be marked by a label and the identifier of that label is to be written in the place of reference. The assumption will take on the form

assume *identifier–of–label* : *sentence* ;

If the sentence which is to be taken as the assumption is in the form of a conjunction, then the assumption may be recorded in the form of a *collective assumption* by replacing the sign `&` by the word **and** and by labelling every constituent of the conjunction. The single assumption:

assume `ΩG = A ∪ B & A is_closed & B is_closed & A ∩ B = ∅G;`

may accordingly be written in the form of a *collective assumption* thus:

assume that `M1: ΩG = A ∪ B` **and** `M2: A is_closed` **and**

M3: B is_closed **and** M4: A ∩ B = ∅𝔊;

A *collective assumption* takes on the form:

assume that *sequence–of–labelled–sentences* ;

A sequence of labelled sentences is a single labelled sentence or several labelled sentences linked together by the connective **and**.

The splitting of a single assumption into a collective one makes it possible to refer separately to every partial assumption.

The assumption from the example given above may also have the following form:

assume x: Ω𝔊 = A ∪ B & A is_closed & B is_closed;
assume xx: A ∩ B = ∅𝔊;

Since the semantic correlates of the sentences in the form:

α & β & γ and (α & β) & γ and α & (β & γ)

are the same hence the following assumption is correct, too:

assume y: (Ω𝔊 = A ∪ B & A is_closed) & B is_closed;
assume yy: A ∩ B = ∅𝔊;

the same applies to the following one:

assume z: Ω𝔊 = A ∪ B & (A is_closed & B is_closed);
assume zz: A ∩ B = ∅𝔊;

The last two forms of the assumption are least legible, and this is why it is better to use recordings in which superfluous brackets are avoided. But the last two possible forms have been given above in order to show the various recordings.

The formulas in the form

(•) Φ_1 & Φ_2 **implies** Φ_3 and Φ_1 **implies** (Φ_2 **implies** Φ_3)

have one and the same semantic correlate.

Φ_1, Φ_2, Φ_3 are any formulas. If they are implication or equivalences, then in the formulas under (•) brackets should occur in the appropriate places. The same applies to the formulas which will be discussed below.

The semantic correlate of the formula

Φ_1 & Φ_2 **implies** Φ_3

has the following form:

not (($\lceil\Phi_1\rceil$ & $\lceil\Phi_2\rceil$) & $\lceil$**not** $\Phi_3\rceil$)

(where inscription $\lceil\Phi_1\rceil$ denotes the semantic correlates of the formula Φ_1; the same applies, by analogy, to the remaining cases).

It will now be shown how the semantic correlate of the formula Φ_1 **implies** (Φ_2 **implies** Φ_3) is formed. That formula may equivalently be recorded thus:

not (Φ_1 & **not** (Φ_2 **implies** Φ_3))

That formula may be recorded equivalently by making use of the semantic form of implication:

not (Φ_1 & **not not** ($\lceil\Phi_2\rceil$ & $\lceil$**not** $\Phi_3\rceil$))

Next we avail ourselves of the fact that negation is an involution:

not (Φ_1 & ($\lceil\Phi_2\rceil$ & $\lceil$**not** $\Phi_3\rceil$))

Since conjunction is associative the semantic correlate of the above formula may be recorded thus:

not (($\lceil\Phi_1\rceil$ & $\lceil\Phi_2\rceil$) & $\lceil$**not** $\Phi_3\rceil$)

The form thus obtained is also a semantic form of the formula:

(Φ_1 & Φ_2) **implies** Φ_3

The sentence being proved has its antecedent in the form of the following formula:

α_1 & α_2 & α_3 & α_4

where

α_1 stands for Ω𝔊 = A ∪ B
α_2 stands for A is_closed

α_3 stands for `B is_closed`
α_4 stands for `A ∩ B = ∅𝔊`

The consequent of the implication, that is the formula

`A, B are_separated`

will be denoted by γ.

The system, when reading the formula, will add brackets in the appropriate places and transform it into the formula:

$$((\alpha_1 \ \&\ \alpha_2)\ \&\ \alpha_3)\ \&\ \alpha_4$$

Now let the formula $((\alpha_1\ \&\ \alpha_2)\ \&\ \alpha_3)$ be denoted by β. On substituting β in the preceding formula we obtain

$$\beta\ \&\ \alpha_4$$

It follows from earlier analyses that the semantic form of the formulas:

$\beta\ \&\ \alpha_4$ **implies** γ and β **implies** $(\alpha_4$ **implies** $\gamma)$

is the same.

Hence the skeleton of the proof of the thesis

$\beta\ \&\ \alpha_4$ **implies** γ

is the same as the skeleton of the proof of the thesis

β **implies** $(\alpha_4$ **implies** $\gamma)$

Since the sentence

```
Ω𝔊 = A ∪ B & A is_closed & B is_closed & A ∩ B = ∅𝔊
        implies A, B are_separated
```

is processed by the system in the same way as the sentence

```
Ω𝔊 = A ∪ B & A is_closed & B is_closed
    implies (A ∩ B = ∅𝔊 implies A, B are_separated)
```

the skeleton of the proof of the thesis

```
Ω𝔊 = A ∪ B & A is_closed & B is_closed & A ∩ B = ∅𝔊
        implies A, B are_separated
```

may have the form:

```
proof
    ......
    assume q:  Ω𝔊 = A ∪ B & A is_closed & B is_closed;
(now the thesis is: A∩B = ∅𝔊 implies A, B are_separated)
    ......
    assume p:  A ∩ B = ∅𝔊;
(now the thesis is: A, B are_separated)
    ......
    thus A, B are_separated;
    ......
end;
```

The proof of the thesis discussed in this example is presented by the example No.9 in the file `art.lst`.

Remark: The skeleton of the proof of the sentence $\alpha\ \&\ \beta$ **implies** γ *may be as below:*

```
proof
    .....
    assume α;
    .....
    assume β;
    .....
    thus γ;
    .....
end;
```

On the contrary, it cannot be the skeleton of the proof of the sentence

$$\beta \ \texttt{\&} \ \alpha \ \textbf{implies} \ \gamma$$

because in that case a different order of assumptions is required.

3. When proving the *EQUIVALENCE*

$$\alpha \ \textbf{iff} \ \beta$$

one has to prove two implications:

$$\alpha \ \textbf{implies} \ \beta \qquad \text{and} \qquad \beta \ \textbf{implies} \ \alpha$$

Since the sentences

α **iff** β and (α **implies** β) & (β **implies** α)

have one and the same semantic correlate the skeleton of the proof for equivalence is subsumed under the skeleton of the proof of a thesis which is a conjunction.

Here is the skeleton of the proof of equivalence:

α **iff** β
proof
......
thus α **implies** β;
(now the thesis is: α **implies** *β)*
......
thus α **implies** β;
......
end;

Since for Mizar the commutativity of conjunction is not self-evident the order in which the implications are indicated must be as above.

The example No.10 in the file `art.lst` shows a proof of equivalence. The skeleton of that proof is in the same form as above.

The skeleton of the proof of the equivalence α **iff** β may be such as below:

proof
......
thus α **implies** β;
(now the thesis is: β **implies** *α)*
......
assume β;
(now the thesis is: α)
......
thus α;
......
end;

In the example No.11 in the annex there is the proof of the thesis from the example No.10. That proof is carried out in a different way than in the example No.10, which is to say that the skeleton of the proof is different. It is in the form as above.

A justification of an equivalence is also to be found in the example No.30 in the file `art.lst`, where the equivalence is justified straightforwardly (by **by**). The list of references consists of labels of two corresponding implications proved earlier.

4. When proving a *THESIS* which is a *DISJUNCTION* it is worth while bearing in mind one more pair of sentences which are processed by the system in the same way. We mean the sentence

$$\alpha \ \textbf{or} \ \beta \qquad \text{and} \qquad \textbf{not} \ \alpha \ \textbf{implies} \ \beta$$

These sentences have the proof of a disjunction it is convenient to assume the negation of the first constituent of the disjunction (α) and to prove the second constituent.

Should we do it conversely by assuming the negation of the second constituent (β) and by proving the first (α), we would prove the thesis

$$\beta \textbf{ or } \alpha$$

But the sentences

$$\alpha \textbf{ or } \beta \qquad \text{and} \qquad \beta \textbf{ or } \alpha$$

have different semantic correlates.

If the thesis is a disjunction of three constituents, then in its proof one has to assume the negation of the first two constituents and to prove the third. This is done in the example below.

EXAMPLE

```
k < n or k = n or n < k
      proof
            assume A: not k < n & k <> n;
```

(The negation of the first two constituents of the disjunction is assumed and it is now the formula `n < k` *which is the thesis)*

```
            then not k <= n by NAT_1:30;
            then n <= k by NAT_1:14;
            hence n < k by REAL_1:57, A;
      end;
```

See the example No.12 in the file `art.lst`.

5. The thesis is in the form of a *GENERAL SENTENCE.*

In such a case the construction of the skeleton of the proof must begin with a generalization. Generalization is used, for instance, in the proofs of general sentences and in the proofs of sentences which can be presented as general ones. Other occurrences of generalization are:

– *diffuse statement,*

– *definition.*

General speaking, generalization is intended to fix certain objects. It accordingly introduces constants at the level of proof.

Generalization is in the form:

let *variables–qualified–implicitly* ;

In view of the diversified forms of the list of qualified variables generalization may take on the form of one of the expressions presented below:

(a) **let** *identifiers–of–variables* ;

Examples: **let** `x;`, **let** `A, B;`

In the generalization of this kind the types of variables which occur in it are not indicated, which can be seen in the examples above. This means that the identifiers of those variables have the respective types given in the reservation of variables.

(b) **let** *variables–qualified–explicitly* ;

Generalization in this form differs from the preceding one in that the types of the variables occurring in it are indicated.

Examples:

```
let x, y be Any;
let P, Q be (Subset of 𝔊), p be Point of 𝔊;
let a, b be Subset of the carrier of Y;
let a be Subset--Family of the carrier of Y;
```

(where `Y` is a certain topological structure).

Instead of **be** one may alternately use **being**, but let the convention be that **be** is used in the construction **let** ... .

(c) **let** *variables–qualified–explicitly* , *variables–qualified–implicitly* ;

Generalization in this form is, generally speaking, a combination of the two preceding ones. Examples:

let `A` **be** `(Subset` **of** $\mathfrak{G}$`), x;`

let `P1, P2` **be** `(Subset` **of** $\mathfrak{G}$`), p, q` **be** `(Point` **of** $\mathfrak{G}$`), x,y,z ;`

It is known that the sentence in the form:

(•) **for** *lists–of–qualified–variables* **holds** Φ_1 **implies** Φ_2

is processed by the system in the same way as the sentence:

(••) **for** *lists–of–qualified–variables* **st** Φ_1 **holds** Φ_2

If the thesis of sentence being proved has the same form as under (•) or (••), then the generalization may be recorded as follows:

let *lists–of–qualified–variables* **such that** *conditions* ;

The condition in such a generalization must be recorded as one labelled sentence or several labelled sentences linked together by the word **and** . For the formulas marked (•) and (••) the conditions may be recorded, for instant, thus:

`W1:` Φ_1`;` or `W1:` Φ_1 **and** `W2:` **not** Φ_2`;`

The use of generalization in the proof will be visible in the discussion of the skeleton of the proof of a general sentence, to be discussed now. For the time being let it be said only that generalization is a *cut down* in the thesis of the universal quantifier. The proof of a general sentence will be discussed by reference to examples.

In each of the examples to be presented below the corresponding reservation of variables and the content of the sentence which requires a proof will be direct, which is important for the construction of the skeleton of the proof. In the case of indirect proofs skeletons look differently, but that case will be discussed later.

Here are the examples announced:

EXAMPLE 1

reserve $\mathfrak{G}$ **for** `TopSpace, x` **for** `Any, P` **for** `Subset` **of** $\mathfrak{G}$`;`

for `x` **holds** `x` $\in$ `Fr P` **implies** `x` $\in$ `(Cl (`P^c`)` $\cap$ `P)` $\cup$ `(Cl P` $\setminus$ `P)`

proof

......

(Since at this point the thesis is a general statement the construction of the skeleton of the proof begins with a generalization)

let `x;`

(the type of the identifier of `x` *is given in the reservation of variables hence it need not be given again. The generalization results in the cutting down of the universal quantifier in the initial thesis, whereby the thesis has become modified. Now the thesis has the form of an implication. When proving an implication directly we assume its antecedent and prove its consequent. Moreover the generalization has introduced the constant* `x` *at the level of the proof).*

......

assume `x` $\in$ `Fr P;`

(Now the formula `x` $\in$ `(Cl (`P^c`)` $\cap$ `P)` $\cup$ `(Cl P` $\setminus$ `P)` *is the thesis.)*

......

thus `x` $\in$ `(Cl (`P^c`)` $\cap$ `P)` $\cup$ `(Cl P` $\setminus$ `P);` *(final conclusion)*

end;

In the generalization, and hence in the proof as a whole, an identifier other than x could have been used because generalization is to apply to the types of the identifiers of the variables occurring in the quantifier formula after the word **for**. The point is that the types of identifiers in the generalization should agree with the types of the identifiers following **for** in the quantified formula.

For instance, if we have a quantified formula in the form:

for `x` **being** `T` **holds** `Φ(x)`

(where `T` *is a type)*
then the generalization may be as follows:

let `y` **be** `T;`

If in the reservation the identifier `y` has been reserved for a type other than `T` or if it has not been at all taken into consideration in that construction, then in the generalization the appropriate type must be indicated. In both cases the type of the variable introduced by a generalization is valid until the end of a given level of reasoning, that is that level at which a given variable was introduced. But if the identifier `y` has been reserved for the type `T`, then the generalization may be as follows: **let** `y;`.

Remark:

The variable introduced by a generalization may be overridden by another generalization, a statement of choice, a statement of a change of type, an exemplification, an existential assumption and local definition of variable.

The expression **assume** `x ∈ Fr P;` is a single assumption which is one of the forms of assumption.

The sentence being proved has for Mizar the same meaning as the sentence:

for `x` **st** `x ∈ Fr P` **holds** `x ∈ (Cl (P`^{c}`) ∩ P) ∪ (Cl P \ P)`

Hence in accordance with the information about the structure of generalization in the case of a thesis which is a formula with a purified quantifier the skeleton of the proof may be abbreviated as follows:

let `x;` **assume** `x ∈ Fr P;`	} **let** `x` **such that** `A: x ∈ Fr P;`

The proof of the sentence discussed in the first example is shown in the annex, file `art.lst`, under No.13.

Now comes another example illustrating the construction of the skeleton of a proof.

EXAMPLE 2

reserve 𝔊 **for** `TopSpace`, `P` **for** `Subset` **of** 𝔊;

$P \subseteq \mathrm{Cl}\ P$

The definitional expansion of this sentence has the following form:

for `x` **being** `Any` **holds** `x ∈ P` **implies** `x ∈ Cl P`

The skeleton of the proof of the sentence `P ⊆ Cl P` may also be the skeleton of the proof of a sentence which is its definitional expansion. This is guaranteed by the joining to the environment of the directive of definitional **definitions** `TARSKI;`.

Here is the skeleton of the proof of the sentence $P \subseteq \mathrm{Cl}\ P$:

proof

......

(Since at this point the sentence which may be expanded into a general sentence is the thesis generalization may be the first element in the skeleton of the proof.)

let `x` **be** `Any`;

(Now it is the implication which is the thesis, and hence the skeleton of the proof may still consist of the assumption of the antecedent.)

......

assume `x ∈ P`;

(Now it is the formula `x ∈ Cl P`*)*

......

thus `x ∈ Cl P`;

......

end;

The full proof is to be found in the example No.14 in the annex.

EXAMPLE 3

reserve 𝔊 **for** `TopSpace`, `P`, `Q` **for** `Subset` **of** 𝔊;

`P is_dense` **implies for** `Q` **holds** `Q ≠ ∅ & Q is_open` **implies** `P ∩ Q ≠ ∅`

proof

......

(Now it is the implication which is the thesis hence the assumption of the antecedent will be the first element of the skeleton of the proof.)

assume `P is_dense`;

(It is a general sentence which is the thesis at this point, and this means that a generalization will be the next element.)

......

let `Q`;

(Now it is the implication which is the thesis, hence we assume its antecedent.)

......

assume `Q ≠ ∅ & Q is_open`;

(it is the formula `P ∩ Q ≠ ∅` *which is the thesis now.)*

......

thus `P ∩ Q ≠ ∅`; *(final conclusion)*

......

end;

The single assumption

assume `Q ≠ ∅ & Q is_open`;

may be recorded equivalently as a collective assumption:

assume that `M1: Q ≠ ∅` **and** `M2: Q is_open`;

or as two single assumptions:

assume `a: Q ≠ ∅`;

......

assume `b: Q is_open`;

The skeleton of the proof of the sentence under consideration may also be as follows:

proof

......

(Now it is the implication which is the thesis hence the assumption of its antecedent will be the first element in the skeleton of the proof.)

assume `P is_dense`;

(At this point a general sentence is the thesis, and this means that a generalization will be the next element.)

......

let `Q` **such that** `Z1: Q ≠ ∅` **and** `Z2: Q is_open`;

(The formula `P ∩ Q ≠ ∅;` *is the thesis.)*

......

thus P ∩ Q ≠ ∅; *(final conclusion)*

......

end;

(See annex, - example No.15.)

EXAMPLE 4

reserve 𝔊 **for** TopSpace;

for 𝔋 **being** SubSpace **of** 𝔊 **for** A **being** Subset **of** 𝔋

holds A **is** Subset **of** 𝔊

Skeleton of the proof:

proof

......

(The general sentence being proved is now the thesis.)

let 𝔋 **be** SubSpace **of** 𝔊;

(The general sentence

for A **being** Subset **of** 𝔋 **holds** A **is** Subset **of** 𝔊

is now the thesis.)

......

let A **be** Subset **of** 𝔋;

(The formula A **is** Subset **of** 𝔊 *is the thesis)*

......

thus A **is** Subset **of** 𝔊;

......

end;

The generalization in the proof above can be recorded more briefly, namely:

let 𝔋 **being** (SubSpace **of** 𝔊), A **being** Subset **of** 𝔋;

This is due to the fact that the sentences

for x **for** y **holds** α and **for** x, y **holds** α

have one and the same semantic correlate.

The proof of the thesis in this example is shown in the example No.16 in the annex.

6. The thesis is an *EXISTENTIAL SENTENCE,*

that is a sentence in the form:

ex *list–of–qualified–variables* **st** *formula*

Let the formula

ex x **being** T **st** Φ(x)

be the thesis.

The proof of this thesis consist in indicating an object of the type T which satisfies the condition Φ(x).

To do so we shall avail ourselves of the construction **take** ... , called *exemplification.* That construction, except for generalization, assumption, and conclusion, modifies the thesis of the proof. While generalization results in the *cutting down* of the universal quantifier in the thesis, exemplification *cuts down* the existential quantifier in the thesis. Exemplification with equalization introduces a constant at the level of the proof, that constant being accessible from the moment of being introduced to the end of that level of reasoning at which it has been introduced, unless it is overridden by another exemplification, a generalization, a statement of choice, a statement of a change of type, an existential assumption, or a local definition of a variable.

Consider, for instance, the theorem

(•) x ∈ Int P **iff** **ex** Q **st** Q is_open & Q ⊆ P & x ∈ Q

The proof of this theorem consist of justifications of two implications. We shall write the skeleton of the proof of the first of them.

```
x ∈ Int P implies ex Q st Q is_open & Q ⊆ P & x ∈ Q
proof
    ......
```

(An implication is the thesis hence we assume its antecedent.)

```
    assume x ∈ Int P;
```

(Now it is an existential sentence which is the thesis. Note that sentence is satisfied for `Q` *equal* `Int P`*.)*

```
    ......
    take Q = Int P;
```

(At this point the formula `Q is_open & Q ⊆ P & x ∈ Q` *is the thesis. By the construction* **take ...** *we have pointed to the object sought. We have to verify whether it satisfied the conditions stated after the word* **st***, that is the thesis now under consideration. Of course, the identifier* `Q` *in the exemplification, and hence in the further proof, may be replaced by any other identifier.)*

```
    ......
    thus Q is_open & Q ⊆ P & x ∈ Q;
    ......
end;
```

The full proof is shown in the example No.10.

For the thesis proved above there may also be other variations of the construction **take** ... ; .

The expression

take `Q = Int P;`

may be replaced by **take** `Int P;`. In such a case the proof will be as shown in the annex – example No.11.

The proof of the other implication which is a part of the thesis marked by the (•) is shown in the example No.10 and in the example No.11. In either example the proof is carried out in a different way.

Here are two skeletons, given by way of example, of the proof of the sentence

```
    ex a st α(x)
(i) proof
    ......
    take y = τ;
    ......
    thus α(y);
    ......
    end;
(ii) proof
    ......
    take τ;
    ......
    thus α(τ);
    ......
    end;
```

(Now τ *is the corresponding term, and* `y`*, any identifier. Any identifier may be substituted for* `y`*.)*

There are sentences in the proofs in which the exemplification consists of several equalizations of terms, which in such a case must be separated be commas from one another.

Let the sentence

ex x **ex** y **st** α(x,y)

be the thesis. The skeleton of the proof might be as follows:

```
proof
   ......
   take x;
   ......
   take y;
   ......
   thus α(x,y);
   ......
end;
```

We applied here exemplification twice but it could have been done only once. Then the skeleton of the proof would be:

```
proof
   ......
   take x,y;
   ......
   thus α(x,y);
   ......
end;
```

The expression **take** x,y; is also an exemplification in the proof of the thesis

ex x,y **st** α(x,y)

But the sentences

ex x **ex** y **st** α(x,y)   and   **ex** x,y **st** α(x,y)

are ready by the system in the same way.

Remark:

The adding in the proof of a statement which does not contribute anything to the proof and such which has some sntactic correlate as **not contradiction**, *i.e.* VERUM, *is not an error. For instance, if in the proof of the thesis from the example No.7 in the annex we write an additional conclusion, then the proof will take on the form such as in the example No.8. Superfluous* **thus thesis** *would be added, but that would not cause an error. In that case* **thesis** *is the formula* **not contradiction** *(*VERUM*). In the processing of the formulas in that proof into semantic correlates* **not contradiction** *as* VERUM *is disregarded. Likewise the addition of* **assume not contradiction** *is not an error for the same reason as above.*

IV.2.2. INDIRECT PROOFS

So far direct proof have been discussed. But indirect proofs can also be carried out in Mizar. What the skeleton of the proof is like in such cases?

If we are to prove a sentence α indirectly, then we may assume the negation of that sentence and to carry out the proof until the point when we arrive at contradiction. The skeleton of the proof for α might be as follows:

```
proof
     ......
     assume not α;
     ......
     thus contradiction;
```

......

end ;

or else

proof

......

assume not α**;**

......

thus thesis ;

......

end ;

(In this case **thesis** means the formula **contradiction**.)

The formula **not** α may, if that is convenient, be replaced by the already negated sentence α. For Mizar that is indifferent.

The word **contradiction** denotes the logical constant *falsehood*. Self-evidently, **not contradiction**, or `VERUM`, denotes the logical constant *truth*. The word **contradiction** is treated by Mizar as a formula. It may occur not only at the end of an indirect proof. Its other occurrences are:

– in Frænkel's terms;

e.g., {`k + 1:` **not contradiction**}

Indirect proof is frequently used when it is an implication which is the thesis:

α **implies** β

In an indirect proof of this implication one has to assume the antecedent of the implication and the negation of its consequent. The assumption may be either single or collective as below:

assume α**;**
assume not β**;**

or

assume α `&` **not** β**;**

or

assume that `S1:` α **and** `S2:` **not** β**;**

The proof is carried on until the point when we arrive at a contradiction, which is manifested by the properly justified statement **thus contradiction**.

EXAMPLE 1

We shall write the skeleton of the proof of the sentence

`P is_open & P is_nowheredense` **implies** `P = ∅`

proof

......

(Now the implication being proved is the thesis. When proving an implication indirectly we assume its antecedent and the negation of its consequent. This assumption will be recorded in the form of a collective assumption.)

assume that `Z1: P is_open` **and** `Z2: P is_nowheredense` **and** `Z3:` `P` $\neq$ `∅`;

(The formula `P` $\neq$ `∅` *is the assumption of the indirect proof. Further steps of the proof must yields a contradiction.)*

......

thus contradiction ;

......

end ;

The example No.17 in the annex illustrates an indirect proof of a thesis which is an implication.

EXAMPLE 2

We shall write the skeleton of an indirect proof of the sentence:

(for `G` **st** `G is_open` **holds** `p` $\in$ `G` **implies** `P` $\cap$ `G` $\neq \emptyset$**)** **implies** `p` $\in$ `Cl P`

proof

......

assume `A0:` **for** `G` **st** `G is_open` **holds** `p` $\in$ `G` **implies** `P` $\cap$ `G` $\neq \emptyset$;

(Now it is the formula `p` $\in$ `Cl P` *which is the thesis. Since the implication in question is being proved indirectly we now have to assume the negation of its consequent.)*

......

assume not `p` $\in$ `Cl P`;

......

thus contradiction;

......

end;

The full proof is shown in the example No.18.

The expression **not** `p` $\in$ `Cl P` may be replaced by the equivalent expression **not thesis**, where the formula **thesis** denotes the formula `p` $\in$ `Cl P`. Then the skeleton of the indirect proof will be as follows:

proof

......

assume `A0:` **for** `G` **st** `G is_open` **holds** `p` $\in$ `G` **implies** `P` $\cap$ `G` $\neq \emptyset$;

......

assume not thesis;

......

thus contradiction;

......

end;

Here is one more skeleton of the proof of the thesis from the example No.2.

proof

......

assume `A0:` **not thesis**;

(The formula **thesis** *denotes here the sentence being proved. Further steps of the proof must be yield a contradiction.)*

......

thus contradiction;

......

end;

For such a form of recording the checking by the system of the correctness of the proof takes more time than in the case of the previous recordings.

The example No.19 shows the proof of the thesis from the example No.18, but the skeleton of the proof of that thesis has the same form as that presented above.

EXAMPLE 3

We shall write the skeleton of the proof of the sentence:

`A is_a_component_of` $\mathfrak{G}$ `& B is_a_component_of` $\mathfrak{G}$

implies `A = B` **or** `A,B are_separated`

proof

......

(The implication being proved is the thesis. We shall prove it directly and hence we assume its antecedent and prove its consequent.)

assume `Z1:` `A is_a_component_of` $\mathfrak{G}$ `& B is_a_component_of` $\mathfrak{G}$;

(Now it is the disjunction `A=B` *or* `A,B are_separated` *which is the thesis. We assume the negation of the first constituent of that disjunction and prove the truth of the second.)*

......

assume `Z2: A` $\neq$ `B;`

(Now it is the sentence `A,B are_separated` *which is the thesis.That sentence is to be proved indirectly and hence we assume its negation.)*

......

assume `Z3:` **not** `A,B are_separated;`

(Further steps of the proof must yield a contradiction.)

......

thus contradiction;

......

end;

The proof of this thesis is to be found in the example No.20 in the annex. Other indirect proofs are shown in the examples Nos.21, 22, 23, 24.

In the proof of the successive Mizar sentence the construction **consider** ... will be used. The role of that construction in the proof consists in the introduction of constants to the level of the proof.

The *statement of choice*, as the construction **consider** ... is called, may take on the form:

(1) **consider** *list–of–qualified–variables* ;

for instance:

consider `x, y;`
consider `A` **being** `Subset` **of** $\mathfrak{G}$, `a` **being** `Any;`
consider `V` **being set,** `P, Q;`

(2) **consider** *list–of–qualified–variables* **such that** *conditions* *justification* ;

The conditions form a single labelled sentence or several labelled sentences linked together by the word **and**. The justification may be by **by** or by **from** that is by schema. The labelling of the sentence(s) occurring in the conditions is due to the fact that after the statement of choice linking is not allowed.

It may be so that the condition in the statement of choice do not require justification. Then the statement of choice will have the form:

(3) **consider** *list–of–qualified–variables* **such that** *conditions* ;

The statement of choice is in such a form when:

• in the justification of the statement of choice we refer solely to the immediately preceding sentence by linking,

• the conditions are accepted by CHECKER without justification, which is to say that we have to do with tautologies of the propositional calculus or with simple laws of the functional calculus.

The example below illustrates the application of the statement of choice.

EXAMPLE 4

Here is the proof of the sentence

`P is_boundary` **iff (for** `Q` **st** `Q` $\subseteq$ `P & Q is_open` **holds** `Q =` $\emptyset$)

proof

thus `P is_boundary` **implies (for** `Q` **st** `Q` $\subseteq$ `P & Q is_open` **holds** `Q =` $\emptyset$)

proof

(Now the implication being proved is the thesis. We assume its antecedent.)

assume `P is_boundary;`

(Now the general sentence **for** `Q` **st** `Q` $\subseteq$ `P & Q is_open` **holds** `Q =` $\emptyset$ *is the thesis.)*

then `P:` P^c `is_dense` **by** `TOPS_1:83;`

let Q;

(Now the implication Q ⊆ P & Q is_open **implies** Q=∅, *which is to be proved indirectly, is the thesis.)*

assume that P1: Q ⊆ P **and** P2: Q is_open **and** P3: Q ≠ ∅;

(The further steps of the proof must yield a contradiction.)

P^c ∩ Q ≠ ∅ **by** TOPS_1:80,P3,P2,P;
then Q ∩ P^c ≠ ∅ **by** BOOLE:66;
hence contradiction by P1,TOPS_1:20;
end;

(Now the following implication is the thesis:
for Q **st** Q ⊆ P & Q is_open **holds** Q = ∅) **implies** P is_boundary*)*

thus (**for** Q **st** Q ⊆ P & Q is_open **holds** Q = ∅) **implies** P is_boundary
proof

(Now the above implication is the thesis.)

assume K: **for** Q **st** Q ⊆ P & Q is_open **holds** Q = ∅;

(Now the formula P is_boundary, *to be proved indirectly, is the thesis.)*

assume not P is_boundary;

(The assumption of the indirect proof. Further steps of the proof must yield a contradiction.)

then not P^c is_dense **by** TOPS_1:83;
then consider C **being** Subset **of** 𝔊 **such that** Q: C ≠ ∅
and Q1: C is_open **and** Q2: P^c ∩ C = ∅ **by** TOPS_1:80;
C ∩ P^c = ∅ **by** Q2,BOOLE:66;
then C ⊆ P **by** TOPS_1:20;
hence contradiction by K,Q,Q1;
end;
end;

(See annex - the example No.21.)

We shall analyse two statements which occur immediately after the assumption of the indirect proof. The first statement is **not** P^c is_dense;.
The theorem TOPS_1:80 formulates the property of a dense set:

P is_dense **iff** (**for** Q **st** Q ≠ ∅ & Q is_open **holds** P ∩ Q ≠ ∅)

But the statement **not** P^c is_dense says that the complement of the set P is not dense. Then by availing ourselves additionally of the thesis TOPS_1:80 we can infer that:

(1) **ex** Q **st** Q ≠ ∅ & Q is_open & P^c ∩ Q = ∅;

Since there is an object which satisfiers the above conditions, in further analysis we may be arbitrary, but its type must agree with the type of the identifier of Q which occurs in (1), which is to say that it must be the type Subset **of** 𝔊. Hence we may write:

consider C **being** Subset **of** 𝔊 **such that** Q: C ≠ ∅
and Q1: C is_open **and** Q2: P^c ∩ C = ∅;

After the statement of choice the type of the identifier of C has been fixed as Subset **of** 𝔊. If that identifier in the reservation of variables had been reserved for another type then the statement of choice has overridden that type. The constant introduced by the statement of choice in accessible from the moment of its introduction to the end of the given level of reasoning, that is that level of reasoning at which the given constant has been introduced.

Remark: The constant introduced by the statement of choice may be overridden by a generalization, another statement of choice, a statement of a change of type, an exemplification, an existential assumption, and a local definition of variable.

Examples of the application of the statement of choice:

(1) **ex** x **st** x ∈ X ∩ Y;
    then consider x **such that** Z: x ∈ X ∩ Y;

(2) X meets Y;
    then consider x **such that** a: x ∈ X **and** b: x ∈ Y **by** BOOLE:15;

Here is the theorem BOOLE:15:
    X meets Y **iff ex** x **st** x ∈ X & X ∈ Y;

(3) X ≠ ∅;
    then consider x **such that** c: x ∈ X **by** BOOLE:1;

Here is the theorem BOOLE:1:
    X = ∅ **iff not ex** x **st** x ∈ X;

The application of the statement of choice in proofs is illustrated by the examples Nos.11, 15, 21, 22, 23, 25 in the annex. Moreover, the example No.35 shows the application of that construction outside a proof, that is outside the reasoning contained between **proof** and **end**;.

IV.2.3. ON A NEW MIZAR CONSTRUCTION

Let the thesis be an implication whose antecedent is an existential sentence. It may be accordingly be a sentence in the form:

(**ex** x **being** T **st** α(x)) **implies** β

The proof of that thesis may be take on the following form:

proof
    assume A: **ex** x **being** T **st** α(x);
    consider y **being** T **such that** Z:α(y) **by** A;
(or **then consider** y **being** T **such that** Z:α(y);*)*
    ......*(proof of β)*
end;

The identifier in the statement of choice may be selected arbitrarily but so that its type should agree with the type of the identifier of x in the assumption, that is with the type of T.

The assumption and the statement of choice may in that case be replaced by an *existential assumption*:

given x **being** T **such that** Z: α(x);

If the thesis is an implication with the antecedent which is an existential statement, then the assumption of the existence of certain objects (by the construction **assume** ...) and the statement of choice justified by that assumption may be replaced by an existential assumption.

The existential assumption may be in the form:

(a) **given** *list–of–qualified–variables* ;

(b) **given** *list–of–qualified–variables* **such that** *conditions* ;

The conditions may form a single labelled sentence or several labelled sentences linked together by the word **and**.

The range of a constant introduced by an existential assumption is the same as the range of a constant introduced by the statement of choice.

Remark:

Linking is not allowed after an existential assumption.

The application of the existential assumption will be illustrated by examples.

EXAMPLE 1

```
reserve 𝔊 for TopSpace, P,Q for (Subset of 𝔊), x for Any;
(ex Q st Q is_open & Q ⊆ P & x ∈ Q) implies x ∈ Int P
    proof
        assume ex Q st Q is_open & Q ⊆ P & x ∈ Q;
        then consider Q such that Z1: Q is_open and
        Z2: Q ⊆ P and Z3: x ∈ Q;
```

(The antecedent of the implication being proved has been assumed and the appropriate statement of choice has been made. Now the formula `x ∈ Int P` *is the thesis. The remaining steps of the proof are shown below.)*

```
        P^c ⊆ Q^c by TOPS_1:15,Z2;
        then Z4:Cl(P^c) ⊆ Cl(Q^c) by TOPS_1:25;
        Q^c is_closed by Z1,TOPS_1:30;
        then Cl(Q^c) = Q^c by PRE_TOPC:52;
        then Cl(Q^c) ⊆ Q^c by Z4;
        then Q^cc ⊆ (Cl(P^c))^c by TOPS_1:15;
        then Q ⊆ (Cl(P^c))^c by TOPS_1:10;
        then Q ⊆ Int P by TOPS_1:42;
        hence thesis by Z3,BOOLE:5;
    end;
```

(See file - example No.11.)

The first two steps of the proof may be replaces by the following existential assumption:

given `Q` **such that** `Z1: Q is_open` **and** `Z2: Q ⊆ P` **and** `Z3: x ∈ Q;`

The proof then assumes the form as in the example No.10 in the file `art.lst`. Other cases of existential assumptions are given in the examples Nos.25, 26, 27, 28 from the file `art.lst`.

Remark: The conditions given in the existential assumption cannot be recorded in the form of an assumption, that is by the word **assume**. *Hence the following recording is incorrect:*

given `Q;` **assume that** `Z1: Q is_open` **and** `Z2: Q ⊆ P` **and** `Z3: x ∈ Q;`

The exercise in the annex – `Z13.lst` – shows the consequences of such an incorrect assumption.

Likewise, the statement of choice:

consider `Q` **such that** `Z1: Q is_open` **and** `Z2: Q ⊆ P` **and** `Z3: x ∈ Q;`

cannot be recorded thus:

consider `Q;` **assume that** `Z1: Q is_open` **and** `Z2: Q ⊆ P` **and** `Z3: x ∈ Q;`

(See annex - `Z12.lst`.)

IV.3. Other Mizar constructions

IV.3.1. ITERATIVE EQUALITY

We shall now discuss the Mizar construction called *iterative equality.* It finds application in proofs of sentences which are equality formulas. Those formulas must satisfy certain conditions, namely they cannot contain free variables. The variables occurring in such formulas must be fixed. They are fixed by generalization, exemplification, statement of choice, statement of a change of type, or local definition of variable.

While *iterative equality* does not introduce any new idea of the proofs.

Let us examine the proof of the theorem $(P^c)^c$ = P, which will later be used to illustrate *iterative equality.*

```
(Pᶜ)ᶜ = P
proof
    (Pᶜ)ᶜ = Ω𝔊 \ (Pᶜ) by TOPS_1:5;
    then (Pᶜ)ᶜ = Ω𝔊 \ (Ω𝔊 \ P) by TOPS_1:5;
    then (Pᶜ)ᶜ = Ω𝔊 ∩ P by BOOLE:82;
    hence (Pᶜ)ᶜ = P by TOPS_1:3;
end;
```

Note the following facts which are characteristic for the thesis and its proof:

1. The sentence proved is an equality formula.
2. The formula occuring in every step of the proof is an equality. Moreover the term on the left side of the equality is the same in each step ($(P^c)^c$).
3. Every step of the reasoning, beginning with the second one, refers to the preceding one (by linking).

These facts suffice for the proof of the thesis $(P^c)^c$ = P to be carried out by an iterative equality, which in the case under consideration has the form:

```
(Pᶜ)ᶜ = Ω𝔊 \ (Pᶜ) by TOPS_1:5
      .= Ω𝔊 \ (Ω𝔊 \ P) by TOPS_1:5
      .= Ω𝔊 ∩ P by BOOLE:82
      .= P by TOPS_1:3;
```

What does a recording mean? For instance, the inference

```
      .= Ω𝔊 \ (Ω𝔊 \ P) by TOPS_1:5
```

is another recording if the expression

```
then (Pᶜ)ᶜ = Ω𝔊 \ (Ω𝔊 \ P) by TOPS_1:5
```

where **then** denotes reference to the preceding sentence, that is the sentence

$$(P^c)^c = \Omega\mathfrak{G} \setminus (P^c)$$

It must be borne in mind that before the symbol .= after the justification of the preceding step of the reasoning, we do not put the semicolon ; . The semicolon is required at the end of the *iterative equality*, that is after the last expression in the form

.= *term* *justification*

The *iterative equality* given above is not the complete proof of the thesis $(P^c)^c$ = P because it lacks the conclusion terminating the proof. It suffices to add:

hence $(P^c)^c$ = P

The word **hence** means that the sentence $(P^c)^c$ = P has been justified by reference to the entire reasoning in the form of an *iterative equality.* Of course, the conclusion may be recorded by means of **thus**, but then, in order to refer to *iterative equality*, we have to prefix the equality formula which opens that equality by a label. We might also write **thus** before the *iterative equality.* Moreover, the conclusion $(P^c)^c$ = P might be replaced by the formula **thesis**. The proof of the thesis $(P^c)^c$ = P may accordingly have the form:

proof

(After the writing of the word **proof** *the variables* 𝔊 *and* P *have been fixed. Since the sentence being proved is read by the system as a quantified formula in the form:* **for** 𝔊, P **holds** $(P^c)^c$ = P, *after the writing of the word* **proof** *the system automatically carries out the generalization* **let** 𝔊, P; *and thus fixes the variables* 𝔊 *and* P. *The variable* 𝔊 *is fixed because* P *has the type* Subset **of** 𝔊.*)*

```
(Pᶜ)ᶜ = Ω𝔊 \ (Pᶜ) by TOPS_1:5
      .= Ω𝔊 \ (Ω𝔊 \ P) by TOPS_1:5
      .= Ω𝔊 ∩ P by BOOLE:82
      .= P by TOPS_1:3;
```

```
        hence thesis;
    end;
```
(See example No.33 in the file `art.lst`.)

Here is another version of the proof of the thesis under consideration:

```
    proof
        A: (P^c)^c = ΩG \ (P^c) by TOPS_1:5;
                  .= ΩG \ (ΩG \ P) by TOPS_1:5
                  .= ΩG ∩ P by BOOLE:82
                  .= P by TOPS_1:3;
        thus thesis by A;
    end;
```

The above proof can be slightly abbreviated if the last two statements:

```
        .= P by TOPS_1:3;
        thus thesis by A;
```

are replaced by a single statement in the following form:

```
        hence (P^c)^c = P by TOPS_1:3;
```

or by the statement

```
        hence thesis by TOPS_1:3;
```

The proof then takes on the form as in the example No.34 in the file `art.lst`.

The various steps of the reasoning in our iterative equality had a straightforward justification (by **by**). In an iterative equality there may also be justifications by schema, but not by proof. In the simplest case, when a given step of the reasoning is self-evident to CHECKER, the justification of that step may be empty.

The sentence $(P^c)^c$ = P can be proved by iterative equality by fixing the variables $\mathfrak{G}$ and P through the statement of choice. In such a case the words **proof** and **end** should not be written. We then may have:

```
        consider G, P;
        (P^c)^c = ΩG \ (P^c) by TOPS_1:5
             .= ΩG \ (ΩG \ P) by TOPS_1:5
             .= ΩG ∩ P by BOOLE:82
             .= P by TOPS_1:3;
```

The variables $\mathfrak{G}$ and P fixed in this way make it possible to use the sentence proved only for such variables as fixed here, that is for P and $\mathfrak{G}$ (see example No.35). Such a way of proving is thus not practical. If the variables are not fixed, then errors will be reported as in the example `Z11.lst` in the annex. The error No.62 states that free variables are not allowed in the iterative equality, and the error No.140, that there is an unknown variable.

Iterative equality can be illustrating as below.

If t_1, t_2, ... ,t_n, t_{n+1} are corresponding terms then the reasoning

$t_1 = t_2$ & $t_2 = t_3$ & ... & $t_n = t_{n+1}$ straightforward-justification hence $t_1 = t_{n+1}$;
may be equivalently replaced by another reasoning, namely the iterative equality in the form:

t_1 = t_2 *straightforward-justification*
  .= t_3 *straightforward-justification*
  ....
  .= t_n *straightforward-justification*
  .= t_{n+1} *straightforward-justification* ;

EXAMPLE

The theorem

```
Int(Int P) = Int P
```

can be proved as below

proof
  Int P = (Cl (P^c))^c **by** TOPS_1:42;
  then Int(Int P) = (Cl (((Cl (P^c))^c)^c))^c &
   (Cl (((Cl (P^c))^c)^c))^c = (Cl (Cl (P^c)))^c &
   (Cl (Cl (P^c)))^c = (Cl (P^c))^c **by** TOPS_1:10,TOPS_1:42,TOPS_1:26;
  hence Int(Int P) = Int P **by** TOPS_1:42;
end;

(See example No.31 in the file art.lst.)
or by reference to *iterative equality*:

proof
  Int P = (Cl (P^c))^c **by** TOPS_1:42;
  then Int(Int P) = (Cl (((Cl (P^c))^c)^c))^c **by** TOPS_1:42
      .= (Cl (Cl (P^c)))^c **by** TOPS_1:10
      .= (Cl (P^c))^c **by** TOPS_1:26;
  hence thesis by TOPS_1:42;
end;

The next example in the file art.lst includes a proof of the sentence Fr P = Fr (P^c) with the use of *iterative equality*.

IV.3.2. DIFFUSE STATEMENT

It is sometimes so that in the proof of a certain thesis it is convenient to justify (an) auxiliary sentence(s). If that sentence cannot be justified straightforwardly (by **by**) or by schema (by **from**), then we have to carry out a proof (i.e., a certain reasoning beginning after the word **proof** and concluded by the word **end**). Then we will get nested proofs. Hence the proof may have lesser clarity. But there is a certain Mizar construction which is applicable in the situation described above. We mean the construction in the form:

now *reasoning* **end**;

The various steps in the reasoning are formed on the same principle as the steps of the proof. The application of this new construction in proofs will be illustrated by an example. Let us consider the thesis

$\mathfrak{F}$ is_closed **implies** meet $\mathfrak{F}$ is_closed

It is a thesis in the form α **implies** β, where
α is the formula $\mathfrak{F}$ is_closed ,
β is the formula meet $\mathfrak{F}$ is_closed .
Moreover, let
γ will be the formula $\mathfrak{F} \neq \emptyset$.

This thesis be proved neither by **by** nor by schema. Hence a proof must be carried out. Since the formula in the form

(γ **implies** β) & (**not** γ **implies** β) **implies** β

is a tautology it is convenient to prove in the proof two auxiliary sentences in the form

γ **implies** β and **not** γ **implies** β

In the proof these sentences are labelled T and K1, respectively.

Here is the proof of the sentence:

(•) $\mathfrak{F}$ is_closed **implies** meet $\mathfrak{F}$ is_closed
proof
  assume $\mathfrak{F}$ is_closed;
  then A: COMPLEMENT($\mathfrak{F}$) is_open **by** TOPS_2:16;

```
T: 𝔉 ≠ ∅ implies meet 𝔉 is_closed
      proof
            assume 𝔉 ≠ ∅;
            then union COMPLEMENT(𝔉) is_open by TOPS_2:26;
            hence meet 𝔉 is_closed by TOPS_1:29,A;
      end;
𝔉 = ∅ implies meet 𝔉 is_closed
      proof
            assume 𝔉 = ∅;
            then meet 𝔉 = ∅ by SETFAM_1:2;
            then meet 𝔉 = ∅(𝔊) by PRE_TOPC:11;
            hence meet 𝔉 is_closed by TOPS_1:22;
      end;
hence thesis by T;
end;
```

In this proof the formula

meet 𝔉 is_closed

is denoted by the word **thesis**.
(See example No.37 in file `art.lst`.)

Remark:

The formulas in the form:

β *and* (γ **implies** β) & (**not** γ **implies** β)

do not have one and the same semantic correlate. Hence the skeleton of the proof of a sentence in the form

β

cannot be subsumed under the skeleton of the proof for a conjunction as in the case of the sentence

(γ **implies** β) & (**not** γ **implies** β).

We shall now prove the same thesis in a similar way (the proof will also consist in justifying the sentences γ **implies** β and **not** γ **implies** β) but the recording of the reasoning will be different.
Here is the thesis:

𝔉 is_closed **implies** meet 𝔉 is_closed

```
proof
      assume 𝔉 is_closed;
      then A: COMPLEMENT(𝔉) is_open by TOPS_2:16;
```

We shall prove γ **implies** β *but that sentence will not be written openly. We shall carry out the reasoning resulting in its justification. In Mizar such a construction begins with the word* **now** *and ends in the expression* **end**; .

```
      T1: now
```

(The word **now** *is followed neither by the semicolon nor by the word* **proof**. *Since the sentence being proved is an implication its antecedent is assumed.)*

```
            assume 𝔉 ≠ ∅;
```

(Below there are further steps of the reasoning leading to the justification of the thesis, which at this point is the formula β*)*

```
            then union COMPLEMENT(𝔉) is_open by TOPS_2:26;
            hence meet 𝔉 is_closed by TOPS_1:29,A;
      end;
```

The above reasoning has proved the sentence γ **implies** β.

In order to be able to refer to it one may, as has been done, place a label before **now**. After the reasoning **now** ... **end;** linking is allowed.

The further proof of the thesis may be as follows:

```
        now
              assume 𝔉 = ∅;
              then meet 𝔉 = ∅ by SETFAM_1:2;
              then meet 𝔉 = ∅(𝔊) by PRE_TOPC:11;
              hence meet 𝔉 is_closed by TOPS_1:22;
        end;
        hence meet 𝔉 is_closed by T1;
  end;
```

(In the justification of the conclusion there occurred the label `T1` *of the previously proved sentence* γ **implies** β*.)*

The above proof is shown in the example No. 38 in the annex.

Since before the word **now** we do not write the thesis to be proved by that construction the construction in the form

now *reasoning* **end;**

has been called *diffuse statement.*

Remark:

1. The thesis proved by diffuse statement is read by analysing the skeleton of that reasoning. The principles of skeletoning in the construction **now** ... *are the same as in proofs. The words* **now** *and* **end** *in a sense replace the words* **proof** *and* **end**, *respectively.*

2. The formula **thesis** *in diffuse statement denotes the thesis of the immediate external proof. The use of the formula* **thesis** *is allowed throughout diffuse statement on the condition that such reasoning is contained in a certain proof (***thesis** *may be used only within a proof).*

3. If we prefix **now** *by a label, then the reference to that label means reference to the thesis proved in diffuse reasoning opening with the word* **now** *preceded by a given label.*

4. Linking is allowed after diffuse statement.

5. It is not allowed to write **then**, **thus**, **hence** *before* **now**.

6. Every reasoning which begins with **now** *must end in* **end**.

EXAMPLE

The sentence in the form

α **or** β **implies** γ

is to be proved. We shall show what the proof of that sentence with the application of a diffuse statement might be. Since the formula

(α **implies** γ) & (β **implies** γ) **implies** (α **or** β **implies** γ)

is a tautology it is worth while making use of the auxiliary sentences

α **implies** γ and β **implies** γ

in the proof. Thus the proof may be as follows:

```
  proof
        ......
        assume P: α or β;
```

(Now γ *is the thesis.)*

```
        ......
        A: now
              ......
              assume α;
              ......
```

thus γ;
......
end;
......
now
......
assume β;
......
thus γ;
......
end;
hence γ **by** `A,P`;
......
end;

Conclusion in the various reasonings may be replaced by **thesis**. In diffuse statements **thesis** would mean γ as does thesis at the end of the proof.

EXAMPLE

When proving a sentence in the form

α **iff** β

it is sometimes convenient to avail oneself of the fact that the formula

(**not** α **implies not** β) **implies** ((**not** β **implies not** α) **implies** (α **iff** β))

is a tautology. In diffuse statement one can prove two auxiliary sentences in the form

not α **implies not** β
not β **implies not** α

As an example we may use the proof of the theorem `CONNSP_1:11` which is:
𝔊 `is_connected` **iff** `for A, B` **being** `Subset` **of** 𝔊 **st** Ω𝔊 `=` A ∪ B `&` A ≠ ∅𝔊 `&` B ≠ ∅ 𝔊 `&` A `is_closed` `&` B `is_closed` **holds** A ∩ B ≠ ∅𝔊

In such a case

α *is the formula* 𝔊 `is_connected`
β *is the formula* **for** `A, B` **being** `Subset t of` 𝔊 **st** Ω𝔊 `=` A ∪ B `&` A ≠ ∅𝔊 `&` B ≠ ∅𝔊 `&` A `is_closed` `&` B `is_closed` **holds** A ∩ B ≠ ∅𝔊
not α *is the formula* **not** 𝔊 `is_connected`
not β *is the formula* **ex** `A, B being Subset` **of** 𝔊 **st** Ω𝔊 `=` A ∪ B `&` A ≠ ∅𝔊 `&` B ≠ ∅𝔊 `&` A `is_closed` `&` B `is_closed` `&` A ∩ B `=` ∅𝔊

In the first diffuse statement we shall prove the sentence

not β **implies not** α

and the second, the sentence

not α **implies not** β

Here is the form of such a proof:

proof
T: **now given** `A, B` **being** `Subset` **of** 𝔊 **such that**
Z1: Ω𝔊 `=` A ∪ B **and**
Z2: A ≠ ∅𝔊 `&` B ≠ ∅𝔊 **and**
Z3: A `is_closed` `&` B `is_closed` `&` A ∩ B `=` ∅𝔊;
......
thus not 𝔊 `is_connected` **by** ... ;

```
        end;
        now assume not 𝔊 is_connected;
              ......
         thus ex A, B being Subset of 𝔊 st Ω𝔊 = A ∪ B & A ≠ ∅𝔊 &
           B ≠ ∅𝔊 & A is_closed & B is_closed & A ∩ B = ∅𝔊 by ... ;
        end;
        hence thesis by T;
    end;
```

The complete proof is given in the example No. 25 in the file `art.lst`.
Other examples of diffuse statements are to be found under Nos. 26, 27, 28, 29 in the file `art.lst`.

IV.3.3. STATEMENT OF A CHANGE OF TYPE

A change of the type of the object under consideration is sometimes necessary in proofs. This is due, amount other things, to the fact that certain theorems are proved only for objects of a definite type. For instance in articles pertaining to the topological space there are theorems, for instance, on points of that space.

We shall write the proof of the sentence

P ⊆ Cl P

It will be a proof by definitional expansion.

```
    reserve 𝔊 for TopSpace, x for Any, P, Q, B for Subset of 𝔊;
    P ⊆ Cl P
        proof
            let x; assume x:  x ∈ P;
```

(Now the formula x ∈ Cl P *is the thesis.)*

Should we prove the sentence

(•) **for** B **being** Subset **of** 𝔊 **st** B is_closed **holds**
  A ⊆ B **implies** p ∈ B

we could obtain p ∈ Cl P from the theorem PRE_TOPC:45. But if that theorem is to be applied the indicator of x, which has the type Any, must be treated as a point of the topological space 𝔊, that is as an object whose type is Point **of** 𝔊.

To do so we shall avail ourselves of the Mizar construction in the form:

reconsider *list–of–changes–of–type* **as** *type* **of** *justification* ;

By using this construction in the proof under consideration we may write:

reconsider t = x **as** Point **of** 𝔊 **by** TOPS_1:1,x;

which means:

let us consider x as a point of the topological space 𝔊

(where t is any identifier).

Remark: In the equalization t = x *the identifier of the object whose type is being changed must be on the right side of the equality. On the left side there may be any identifier, which need not be drawn from the list in the reservation of variables.*

The *statemant of a change of type* – as the construction **reconsider** ... is called – results in the fact that in the further part of the present level of reasoning (the level at which the constant has been introduced) the type of the identifier of x, if not given explicitly, will be Point **of** 𝔊 even though in the reservation the identifier of x was reserved for the type Any. Of course, the change of type must be properly justified. In our case we have to refer to the theorem TOPS_1:1, which is:

x ∈ P **implies** x **is** Point **of** 𝔊 and to the assumption, that is the formula x ∈ P.

The next step in the present proof consists in the justification of the sentence marked (•). Then we have only to write the conclusion of the main proof. Here is the completion of the main proof:

```
     for B being Subset of 𝔊 st B is_closed holds P ⊆ B implies t ∈ B
               proof
                    let Q; assume Q is_closed; assume P ⊆ Q;
                    hence t ∈ Q by x,BOOLE:11;
               end;
          hence x ∈ Cl P by PRE_TOPC:45;
     end;
```

(See example No. 14 in the file `art.lst`.)

The statement of a change of type in the proof under consideration may be recorded otherwise than in the form of equalization.

If we want to change the type of the identifier of x, then that fact may be recorded thus:

```
               reconsider x as Point of 𝔊 by TOPS_1:1, x;
```

We have accordingly to change, in the previous version of the proof, the identifier of t in all its occurrences into the identifier of x. The proof then takes on the form:

```
   P ⊆ Cl P
  proof
     let x; assume x:  x ∈ P;
     reconsider x as Point of 𝔊 by TOPS_1:1, x;
     for B being Subset of 𝔊 st B is_closed holds P ⊆ B implies x ∈ B
        proof
             let Q; assume Q is_closed; assume P ⊆ Q;
             hence x ∈ Q by x,BOOLE:11;
        end;
     hence thesis by PRE_TOPC:45;
  end;
```

In the last statement **thesis** could not have been replaced by x ∈ Cl P because that sentences says nothing about x from the generalization, but refers to x from reconsider (reconsider has overridden the generalization).

Example No. 13 also contains a statement of a change of type.

The constant introduced by a statement of a change of type may be overridden by generalization, statement of choice, another statement of a change of type, exemplification, existential assumption, and local definition of variable.

The list of changes of type may have the form of several equalizations (or terms), which in such a case must be separated by commas from one another.

BIBLIOGRAPHY

[1] Selected papers on Mizar and their abstracts:
- [PRE_TOPC] Beata Padlewska and Agata Darmochwał. Topological Spaces and Continuous Functions. *Formalized Mathematics*, 1(**1**): 223–230, 1990.
- [TOPS_1] Mirosław Wysocki and Agata Darmochwał. Subsets of Topological Spaces. *Formalized Mathematics*, 1(**1**): 231–237, 1990.
- [CONNSP_1] Beata Padlewska. Connected Spaces. *Formalized Mathematics*, 1(**1**): 239–244, 1990.

[2] Rasiowa, H., Introduction to Modern Mathematics, Amsterdam – London – Warsaw 1973.

[3] Pogorzelski, W.A., Słupecki, J., O dowodzie matematycznym (Mathematicals Proofs), Warsaw 1970.

[4] Pogorzelski, W.A., Klasyczny rachunek kwantyfikatorów (The Classical Functional Calculus), Warsaw 1970.

[5] Engelking, R., Sieklucki, K., Wstęp do topologii (Introduction to Topology), Warsaw 1986.

[6] Trybulec, A., Rudnicki, P., A Collection of TEXed Mizar Abstracts, University of Alberta, Canada, 1989.

[7] Trybulec, A., Syntaktyka Mizara (Mizar Syntactics), Białystok 1989.

Annex