A Collection of TEXed Mizar Abstracts[1]

Piotr Rudnicki[2]

Dept. of Computing Science
The University of Alberta
Edmonton, Alberta
Canada T6J 2H1

Andrzej Trybulec[3]

Institute of Mathematics
Warsaw University
15-276 Białystok
Poland

June 19, 1989

[1]This text has been distributed as a Technical Report TR 89-18. An attempt to reconstruct the electronic version of the text in November 2001 from 1989 sources resulted in a slightly different layout of pages.

[2]Supported in part by NSERC Grant No. OGP 9207.

[3]Supported in part by grant RPBP.III-24.B1 from The Polish Ministry of National Education and NSERC Grant No. OGP 9207. This work has been done while the author visited The University of Alberta in Spring 1989.

Abstract

We report our work on increasing readability of mathematical texts used as input to theorem verifiers such as Mizar. Even though the source Mizar text is written in extended ASCII (256 characters), it lacks the power of symbolic expression needed for mathematical texts. In our work, the source Mizar texts were automatically translated into TEX input. The translation was done at a primitive level and was restricted to the lexical structure of the source texts. We briefly describe the technology of TEXing and attach TEXed abstracts of 31 Mizar articles written by 12 authors. The results of the experiments are encouraging and the work on TEXing full Mizar articles will be continued. The main conclusion of our work is that the quality typesetting of Mizar texts requires full syntactic analysis including treatment of some contextual dependeces.

Contents

Chapter 1

Introduction

1.1 Motivation

The idea that an automatic device should check our logical derivations is by no means new. It can be traced back not only to Pascal and Leibnitz, but to Ramon Llull. In recent years, several projects have aimed at providing computer assistance for doing mathematics. Among the better known there are: Nuprl [1], THEAX [7], AUTOMATH [2], EKL [3], QUIP [12]. The specific goals of these projects vary, however, they have one common feature: the human writes mathematical texts and the machine verifies their correctness.

The input to any of such systems is an ASCII (or some other code) file. As such it can be printed or seen at a display monitor. However, the input texts are meant to be readable for the computer (taking into account current input devices) and they are visually far from what one would call a mathematical text (even if their semantic contents fully justifies the name). In consequence, the human readers are reluctant to read the texts, although their authors did not mean only computers as potential readers. We report our work on increasing readability of mathematical texts used as input to theorem verifiers.

The system we have experimented with is Mizar [13]. The Mizar input text is written in extended ASCII. The following is an example of a theorem in such a text:

```
:: FUNCT_1:159
   f is_one-to-one iff for y ex x st f"{y} c= {x}
;
```

Our goal was to make this text better looking by processing it automatically. Here is what we have obtained:

Theorem FUNCT_1:159. f is 1-1 **iff for** y **ex** x **st** $f^{-1}\{y\} \subseteq \{x\}$.

The printouts included in this report have been obtained using TEX[4] and LATEX[5]. However, we wanted that neither the author of the Mizar text nor the reader of the text ever sees the TEX input. The TEX input generated automatically in our experiment for the above example is as follows:

```
Theorem FUNCT\_1:159. f
{\sf is 1-1}
{\bf iff}
{\bf for}
y
{\bf ex}
x
{\bf st}
f$^{-1}$\{y\}
$\subseteq$
\{x\}\vspace{1mm}.
```

We have prepared a set of software tools that convert the Mizar source text into the TEX input. Our experiment was limited in the sense that we generate the TEX input after doing only the lexical analysis of the Mizar text.

Our original goal was to obtain a readable printout of these Mizar texts that we needed to look through to write our new article (not included in this collection). Working with TEX was such a fun that we have ended up processing all Mizar articles available to us. We hope that the contents of this report will be useful as a reference for other Mizar users.

1.2 The PC Mizar system

1.2.1 A bit of history

The project Mizar started in 1975 in Poland under the leadership of Andrzej Trybulec. Its original goal was to design and implement a software environment to assist the process of preparing mathematical papers.

After several years of experiments, a language called Mizar 2 had been designed (by A. Trybulec) and implemented on ICL 1900 (by Cz. Byliński, H. Oryszczyszyn, P. Rudnicki, and A. Trybulec, 1981). The system was later ported to other computers (mainframe IBM and also to UNIX). It has included the following features: structured types, type hierarchy, comprehensive definitional facilities, built-in fragments of arithmetics, and built-in variant of set theory. Among other works with Mizar 2, there was an attempt to prove properties of programs in it [11].

The Mizar team effort in the following years resulted in developing other Mizar languages and their implementations but their character was experimental (Mizar 3, Mizar HPF); the systems were not distributed outside the Mizar group in Białystok. There was one exception. A subset of Mizar, named Mizar MSE, was implemented (by R. Matuszewski, P. Rudnicki, and A. Trybulec) in 1982 and has been widely used since then. The system is meant for teaching elementary logic with stress on the practical aspects of constructing proofs. The Mizar MSE language encompasses many sorted predicate calculus with equality. However, the language does not support functional notation. There are numerous implementations of Mizar MSE, see [15, 14, 6, 10, 9, 8]

In 1986 Mizar 4 was implemented as a redesign of Mizar 2 and distributed to several dozen users. Each Mizar 4 article included the preliminaries part where the author could state some axioms that were not checked for validity. In 1988 the design process of the language was completed (by A. Trybulec) and this language is named simply Mizar. While articles in Mizar 4 must be self-contained, Mizar allows for cross-references among articles. Moreover, an author of a Mizar text is not allowed to introduce new axioms. Only the predefined axioms can be used, everything else must be proved.

Recently, the main effort in the Mizar project has been in building the library of Mizar articles.

1.2.2 The overall structure

In this subsection we give a brief overview of PC Mizar, further subsections elaborate on some aspects that are relevant to this report. PC Mizar is a Mizar processor implemented on IBM PCs under DOS (by Cz. Byliński, A. Trybulec, and S. Żukowski from Warsaw University in Białystok).

The central concept of Mizar is a *Mizar article*. Such an article can be viewed as an extremely detailed mathematical text written in a fixed formal notation. The source text of a Mizar article is prepared as a text file (its name has obligatory extension `.miz`). There are rather few interesting things that one can prove in a short Mizar article without making references to other articles. Usually, we base our work on the achievements of others.

The power of the Mizar system is in automatic processing of cross-references among articles. This is done by maintaining a Mizar library. The library consists of files that are automatically created from source Mizar articles and it also includes vocabulary files. The vocabulary files (extensions `.voc` and `.pri`) exist separately from library articles. They contain declarations of symbols that can be included into the lexical environment of an article.

The Mizar processor is a program that verifies the correctness of Mizar articles. To verify an article, the program must run in the appropriate software environment. Namely, it must have access to all the vocabulary and library files referenced from the given article. PC Mizar assumes certain organization of directories in which the vocabulary and library files are kept (we will not discuss it here).

Five library files are created in the process of including an article into the Mizar library. These are:

- *format file* (extension `.nfr`) that, for each constructor (e.g. function) introduced in the article, gives certain information that is used during parsing.
- *signature file* (extension `.sgn`) that, for each constructor, specifies types of its arguments and some additional information, e.g. the type of the result of a function.

- *definitions file* (extension `.def`) for each definition from the article, the definiens is stored in this file, the definiendum is stored in the signature file.
- *theorems file* (extension `.the`) stores the theorems proved in the article (without proofs).
- *schemes file* (extension `.sch`) stores the schemes proved in the article (without proofs).

The environment part of each article (between `environ` and `begin`) must declare all other PC Mizar units that are referenced from the article.

1.3 The lexical context of an article

The set of symbols that can be used in a Mizar article is not fixed externally. The author of an article indicates which tokens are taken into account while tokenizing the article. By a *lexicon* of an article we mean the set of such tokens. The lexicon of an article consists of the *basic lexicon* and some *additional lexicons*. Additional lexicons are not associated with any single Mizar article, they can be shared by many articles.

The basic lexicon includes the following tokens:

- *Reserved words*:

`and`	`as`	`assume`	`be`
`begin`	`being`	`by`	`case`
`cases`	`coherence`	`compatibility`	`consider`
`consistency`	`contradiction`	`correctness`	`definition`
`definitions`	`end`	`environ`	`ex`
`existence`	`for`	`from`	`func`
`given`	`hence`	`holds`	`if`
`iff`	`implies`	`is`	`it`
`let`	`means`	`mode`	`not`
`now`	`of`	`or`	`otherwise`
`per`	`pred`	`proof`	`provided`
`qua`	`reconsider`	`redefine`	`reserve`
`scheme`	`schemes`	`signature`	`set`
`st`	`struct`	`such`	`take`
`that`	`the`	`then`	`theorem`
`theorems`	`thesis`	`thus`	`uniqueness`
`vocabulary`			

- *Special symbols*:

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

For `(#` and `#)` there are synonymous characters with decimal codes 174 and 175 whose usual graphical representation resembles $\ll$ and $\gg$, respectively.

- *Numerals* are strings of decimal digits.
- *Identifiers* are strings of letters, digits, underscore (_), and apostrophe (') that are not reserved words, symbols, numerals.

The additional lexicons are defined in the *vocabulary* files. An additional lexicon is a set of symbols which are strings of arbitrary characters excluding control characters, space, and double colon. Each line of such a file introduces a symbol. Symbol are grouped into the following classes: mode symbol, function symbol, left or right function bracket, structure symbol, selector symbol, and predicate symbol.

If an additional lexicon defines a symbol represented by a string of characters that otherwise forms an identifier, the symbol overrides the identifier.

The symbols introduced in vocabulary HIDDEN are put into the lexicon of every Mizar article. Symbols from other vocabularies are put into the lexicon of an article with the help of the `vocabulary` directive.

1.3.1 The structure of a Mizar article

Each Mizar article is written as a text file. The general structure of such an article is as follows:

`environ`

Environment

`begin`

Text-Proper

The *Text-Proper* contains new facts with their proofs and definitions of new concepts. The *Environment* declares the items in the Mizar library that can be referenced from the *Text-Proper*. This part consist of a sequence of directives. There is one format of vocabulary directives:

`vocabulary` *Vocabulary-File-Name*;

This directive adds the symbols introduced in the *Vocabulary-File-Name* to the article's lexicon. We say that this directive declares the vocabulary in the article.

There are four kinds of library directives

`signature` *Signature-File-Name*;

`definitions` *Definitions-File-Name*;
`theorems` *Theorems-File-Name*;
`schemes` *Schemes-File-Name*;

The directive `signature` informs the Mizar processor that the article is permitted to use the notation introduced in article *Signature-File-Name*`.miz`. The directive is needed to parse the *Text-Proper*. The remaining three directives allow us to use definitions, theorems, and schemes (e.g. induction scheme) that are defined or proved in another article.

The *Text-Proper* is a sequence of *Text-Item*s, and there are the following kinds of them:

- *Reservation* is used to reserve identifiers for a type. If a variable has an identifier reserved for a type, and no explicit type is stated for the variable, then the variable type defaults to the type for which its identifier was reserved.
- *Definition-Block* is used to define (or redefine) constructors. There are three sorts of constructors: term constructors (functions), formula constructors (predicates), and type constructors (modes).
- *Structure-Definition* introduces new structures. A structure is an entity that consists of a number of fields that are accessed by selectors.
- *Theorem* announces a proposition that can be referenced from other articles.
- *Scheme* also announces a proposition, visible from outside. It contrast to *theorem*, *scheme* is expressed in terms of second-order variables.
- *Auxiliary-Item* introduces objects that are local to the article in which they occur and are not exported to the library files (e.g. lemmas, definitions of local predicates).

The goal of writing an article is to prove some theorems and/or define some new concepts such that the concepts can be referenced by other authors. Before the theorems and definitions are included into the library they must be proved valid and correct. The Mizar article contains proofs of the theorems and justifications of the correctness of the definitions.

1.3.2 Mizar abstracts

Mizar input texts tend to be lengthy as they contain complete proofs in a rather demanding formalism. New articles strongly depend on already existing ones. Therefore, there was a need to provide the authors with a quick reference to the already collected articles. The solution consisted in automatically creating an *abstract* for each Mizar article. An abstract of an article includes all the items that can be referenced from other articles. Therefore, there is no need to examine the entire article to make a reference to a single theorem. Grammar of PC Mizar abstracts is given in appendix B.

The environment of an abstract contains only the directives for accessing vocabularies and signatures. Figure 1.1 presents an example of such an environment.

```
environ

    vocabulary Boole;
    vocabulary Fam_op;
    vocabulary Sub_op;
    vocabulary Sfamily;

    signature Tarski;
    signature Boole;
    signature Enumset1;
    signature Subset_1;

begin
```

Figure 1.1: Sample environment.

1.3.3 Mizar library

The Mizar group at the Warsaw University (Institute of Mathematics in Białystok) started collecting Mizar articles and organizing them into a library that is distributed to other Mizar users. This report contains the abstracts of the articles in the library as of May 10, 1989. The articles were authored by 12 people.

The person responsible for the library (E. Woronowicz) requires that authors of contributed articles supply an additional file that describes the bibliographic data of the article, a file with extension `.bib`. These files have been processed by us to obtain the title, authors' names, and the summary. They are printed at the beginning of each abstract.

1.4 The technology of TEXing

In our experiment, we have tried to produce a quality output on a laser printer doing only lexical analysis of the source of Mizar abstracts.

1.4.1 Preprocessing

The TEXing of Mizar abstracts was done under UNIX BSD 4.3. The Mizar source files, in extended ASCII IBM Set II, were transferred from IBM PC to UNIX (using `kermit`).

The version of `lex` that we used recognized only first 128 characters of the code. Therefore, we had to do something with the remaining 128 characters. In Mizar PC all these characters can be used in user-defined vocabularies. Every character with code greater than 127 was translated into its 3 digit decimal representation prepended with a backslash.

1.4.2 Lexical analysis

We used `lex` for analysis of Mizar abstracts and the generation of TEX input. Our first attempt to write one `lex` program that would handle all the symbols from vocabularies failed. We have exceeded the capacity of an internal parameter of `lex` that cannot be controlled from outside (number of positions in a state). An attempt to have just a small number of `lex` programs that could process all the abstracts failed because of the prohibitively high running time of `lex` (more than 15 minutes which was too much for us). But this solution had to be abandoned for another and much more serious reason. Namely, if a vocabulary is declared in an article then no symbol from the vocabulary can be used as an identifier, even if it has the syntax of an identifier. E.g. if vocabulary `Boole` is declared in an article then capital `U` cannot be used as an identifier in the article. (The symbol was meant to denote set union.) However, in articles that do not use the vocabulary, `U` is a legal identifier. Therefore, depending on the vocabularies declared in an article `U` is printed either as $\cup$ or as U.

Because of all that, we needed a separate `lex` program for each of the articles. Therefore, we prepared a separate set of `lex` rules for each vocabulary, each kept in a separate file and prepared by hand. The `lex` program for an article is obtained by the catenation of a common beginning part, the files containing rules for vocabularies used in the article, and a common ending part containing rules for Mizar defined symbols. All Mizar reserved words are printed in **boldface**.

1.4.3 Syntax changes

The environment section of an abstract is automatically converted to a different form. The way how it is done can be easily guessed from the text in figure 1.2 that is the printed version of the environment part listed in figure 1.1:

> The symbols used in this article are introduced in the following vocabularies: BOOLE, FAM_OP, SUB_OP, and SFAMILY. The terminology and notation used in this article have been introduced in the following articles: TARSKI, BOOLE, ENUMSET1, and SUBSET_1.

Figure 1.2: TEXed environment.

Some other changes were minor.

- Semicolon was replaced by a period.

- Each theorem starts with the word 'Theorem' followed by a pattern of library reference to it.

- The definition starts with the word 'Definition' and the matching `end` is not printed, indentation is used to improve readability.

1.4.4 Lexem categories and horizontal spacing

For the horizontal spacing all tokens have been classified into 8 groups.

1. Left delimiters: special symbols `(` { `[` `(#` and vocabulary symbols classified as *Left-Function-Bracket*,

2. Right delimiters: special symbols `)` } `]` `#)` and vocabulary symbols classified as *Right-Function-Bracket*,

3. Punctuation marks: special symbols `;` `,` :.

4. Identifiers.

5. Identifier-like symbols: Mizar reserved words and vocabulary symbols that are printed as sequences of letters and possibly some other characters (e.g. the function symbol `the_left_argument_of`).

6. Binary operations: function symbols used in infix notation and printed as one symbol.

7. Prefix operations: function symbols used in prefix notation and printed as one symbol.

8. Postfix operations: function symbols used in postfix notation and printed as one symbol.

For every pair of symbols, we defined the spacing between them depending on their classes. The array in figure 1.3 specifies the spacing rules. The class `0` in the array denotes a special class: beginning of a line, no previous symbol. The meaning of the entries in the array is as follows:

- `0` - no spacing, linebreak not allowed,

- `1` - a regular space,

- `2` - no spacing, linebreak allowed (`linebreak[0]`).

```
                /* 0  1  2  3  4  5  6  7  8 */
int       SPACES [9] [9] = {
      /* 0 */   {  0, 0, 0, 0, 0, 0, 0, 0, 0 },
      /* 1 */   {  0, 0, 0, 0, 0, 0, 0, 0, 0 },
      /* 2 */   {  0, 2, 0, 0, 2, 1, 2, 2, 0 },
      /* 3 */   {  0, 1, 2, 0, 1, 1, 0, 1, 0 },
      /* 4 */   {  0, 0, 0, 0, 1, 1, 0, 0, 0 },
      /* 5 */   {  0, 1, 0, 0, 1, 1, 0, 1, 0 },
      /* 6 */   {  0, 2, 0, 0, 0, 0, 0, 2, 0 },
      /* 7 */   {  0, 0, 0, 0, 0, 0, 0, 0, 0 },
      /* 8 */   {  0, 0, 0, 0, 0, 1, 2, 0, 0 }
        };
```

Figure 1.3: Spacing rules.

1.4.5 Mishaps

In our experiment the analysis of Mizar source texts was limited to lexical analysis only. Mizar vocabularies classify all symbols introduced in them into classes specified in section 1.3. This classification alone is not sufficient to solve some problems, e.g. is a given symbol a symbol of a prefix or an infix operation? Moreover, the same function symbol can be used in the same article as a postfix, prefix, or infix operation. However, without doing syntactic analysis we have no way of guessing which of the three is used in a specific case. Fortunately, the authors of the papers in question did not use this possibility, with some exceptions. E.g. in chapter 10 the author uses the symbol `"`, which is TEXed as superscript $^{-1}$, as a function symbol for three different functions as follows.

- (infix notation) inverse image of a set under a mapping, e.g. $\mathrm{f}^{-1}\mathrm{X}$,
- (postfix notation) inverse of a bijective mapping: e.g. f^{-1},
- (prefix notation) the function induced by a function f on the power set of its range that assigns to a set its inverse image under f: $^{-1}\mathrm{f}$.

Originally, the symbol `"` has been introduced in vocabulary REAL_1 while preparing article REAL_1 and was used as a postfix function to denote the inverse of a real number.

Despite that we used the set of `amssymbols` in LATEX, the symbol for symmetric difference ($\dot{-}$) had to be typeset by hand.

There is also one thing to mention about Polish characters available in TEX. Namely, there is Polish ł as a separate object; some Polish letters can be obtained using accents. However, some Polish letters cannot be constructed using the available features, e.g. ę which was obtained by hand and only poorly resembles the actual character (we did not have time to design a new font).

1.5 Conclusions

We feel that our limited experiment was encouraging. The TEXed texts are much easier to read than the Mizar sources and at the same time visually close enough to the sources. We did not expect that doing only lexical analysis we can obtain the text that looks so well. We also feel that obtaining a better output would require a considerably bigger effort.

The following remarks will be considered in the future work on typesetting of Mizar articles and their abstracts:

- The quality typesetting of Mizar texts requires full syntactic analysis. Moreover, we feel that pure context-free parsing is insufficient, and contextual dependencies must be taken into account. Only in this case we will be able to benefit from the power of the TEX math-mode.
- The authors of Mizar vocabularies should prepare the TEX version of symbols they introduce.
- It seems useful to prepare a set of TEX macros that are specialized for Mizar texts.
- In the future, pre-editing and post-editing during the typesetting seems the only way to solve certain problems.

Acknowledgements

Our thanks are to Włodek Dobosiewicz and Paweł Gburzyński for many helpful comments while writing this report.

Chapter 2

TARSKI

Tarski Grothendieck Set Theory

by

Andrzej Trybulec [1]

Warsaw University (Białystok)

Summary. This is the first part of the axiomatics of the Mizar system. It includes the axioms of the Tarski-Grothendieck set theory. They are: the axiom stating that everything is a set, the extensionality axiom, the definitional axiom of the singleton, the definitional axiom of the pair, the definitional axiom of the union of a family of sets, the definitional axiom of the boolean (the power set) of a set, the regularity axiom, the definitional axiom of the ordered pair, the Tarski's axiom A (the existence of arbitrary large strongly inaccessible cardinals). Also, the definition of equinumerosity is introduced.

The symbols used in this article are introduced in the following vocabularies: EQUI_REL, BOOLE, and FAM_OP.

reserve x, y, z, u **for** Any, N, M, X, Y, Z **for** set.

Theorem TARSKI:1. x **is** set.

Theorem TARSKI:2. (**for** x **holds** $x \in X$ **iff** $x \in Y$) **implies** $X = Y$.

Definition

let y.

func $\{y\} \rightarrow$ set **means** $x \in$ **it** **iff** $x = y$.

[1]Supported by RPBP.III-24.B1.

let z.

func $\{y, z\} \rightarrow$ set **means** $x \in$ **it** **iff** $x = y$ **or** $x = z$.

Theorem TARSKI:3. $X = \{y\}$ **iff** **for** x **holds** $x \in X$ **iff** $x = y$.

Theorem TARSKI:4. $X = \{y, z\}$ **iff** **for** x **holds** $x \in X$ **iff** $x = y$ **or** $x = z$.

Definition

let X, Y.

pred $X \subseteq Y$ **means** $x \in X$ **implies** $x \in Y$.

Definition

let X.

func $\bigcup X \rightarrow$ set **means** $x \in$ **it** **iff** **ex** Y **st** $x \in Y$ & $Y \in X$.

Theorem TARSKI:5. $X = \bigcup Y$ **iff** **for** x **holds** $x \in X$ **iff** **ex** Z **st** $x \in Z$ & $Z \in Y$.

Theorem TARSKI:6. $X =$ bool Y **iff** **for** Z **holds** $Z \in X$ **iff** $Z \subseteq Y$.

Theorem TARSKI:7. $x \in X$ **implies** **ex** Y **st** $Y \in X$ & **not** **ex** x **st** $x \in X$ & $x \in Y$.

scheme Fraenkel{A() $\rightarrow$ set, P[Any, Any]}: **ex** X **st** **for** x **holds** $x \in X$ **iff** **ex** y **st** $y \in A()$ & P[y, x] **provided** **for** x, y, z **st** P[x, y] & P[x, z] **holds** $y = z$.

Definition

let x, y.

func [x, y] **means** **it** $= \{\{x, y\}, \{x\}\}$.

Theorem TARSKI:8. $[x, y] = \{\{x, y\}, \{x\}\}$.

Definition

let X, Y.

pred $X \approx Y$ **means** **ex** Z **st** (**for** x **st** $x \in X$ **ex** y **st** $y \in Y$ & $[x, y] \in Z$) & (**for** y **st** $y \in Y$ **ex** x **st** $x \in X$ & $[x, y] \in Z$) & **for** x, y, z, u **st** $[x, y] \in Z$ & $[z, u] \in Z$ **holds** $x = z$ **iff** $y = u$.

Theorem TARSKI:9. **ex** M **st** $N \in M$ & (**for** X, Y **holds** $X \in M$ & $Y \subseteq X$ **implies** $Y \in M$) & (**for** X **holds** $X \in M$ **implies** bool $X \in M$) & (**for** X **holds** $X \subseteq M$ **implies** $X \approx M$ **or** $X \in M$).

Chapter 3

AXIOMS

Axioms about Built-in Concepts

by

Andrzej Trybulec [1]

Warsaw University (Białystok)

Summary. This abstract contains the second part of the axiomatics of the Mizar system (the first part is in abstract TARSKI). The axioms listed here characterize the Mizar built-in concepts that are introduced in abstract HIDDEN which is automatically attached to every Mizar article. We give definitional axioms of the following concepts: element, subset, Cartesian product, domain (non empty subset), subdomain (non empty subset of a domain), set domain (domain consisting of sets). Axioms of strong arithmetics of real numbers are also included.

The symbols used in this article are introduced in vocabulary BOOLE. The terminology and notation used here have been introduced in article TARSKI.

reserve x, y, z **for** Any, X, X1, X2, X3, X4, Y **for** set.

Theorem AXIOMS:1. (**ex** x **st** x ∈ X) **implies** (x **is** Element **of** X **iff** x ∈ X).

Theorem AXIOMS:2. X **is** Subset **of** Y **iff** X ⊆ Y.

Theorem AXIOMS:3. z ∈ ⟦X, Y⟧ **iff** **ex** x, y **st** x ∈ X & y ∈ Y & z = [x, y].

Theorem AXIOMS:4. X **is** DOMAIN **iff** **ex** x **st** x ∈ X.

Theorem AXIOMS:5. ⟦X1, X2, X3⟧ = ⟦⟦X1, X2⟧, X3⟧.

[1]Supported by RPBP.III-24.B1.

Theorem AXIOMS:6. $[\![X1, X2, X3, X4]\!] = [\![[\![X1, X2, X3]\!], X4]\!]$.

reserve D1, D2, D3, D4 **for** DOMAIN.

Theorem AXIOMS:7. **for** X **being** Element **of** $[\![D1, D2]\!]$ **holds** X **is** TUPLE **of** D1, D2.

Theorem AXIOMS:8. **for** X **being** Element **of** $[\![D1, D2, D3]\!]$ **holds** X **is** TUPLE **of** D1, D2, D3.

Theorem AXIOMS:9. **for** X **being** Element **of** $[\![D1, D2, D3, D4]\!]$ **holds** X **is** TUPLE **of** D1, D2, D3, D4.

reserve D **for** DOMAIN.

Theorem AXIOMS:10. D1 **is** SUBDOMAIN **of** D2 **iff** D1 $\subseteq$ D2.

Theorem AXIOMS:11. D **is** SET DOMAIN.

reserve x, y, z **for** Element **of** REAL.

Theorem AXIOMS:12. x+y = y+x.

Theorem AXIOMS:13. x+(y+z) = (x+y)+z.

Theorem AXIOMS:14. x+0 = x.

Theorem AXIOMS:15. x·y = y·x.

Theorem AXIOMS:16. x·(y·z) = (x·y)·z.

Theorem AXIOMS:17. x·1 = x.

Theorem AXIOMS:18. x·(y+z) = x·y+x·z.

Theorem AXIOMS:19. **ex** y **st** x+y = 0.

Theorem AXIOMS:20. x $\neq$ 0 **implies** **ex** y **st** x·y = 1.

Theorem AXIOMS:21. x $\leqslant$ y & y $\leqslant$ x **implies** x = y.

Theorem AXIOMS:22. x $\leqslant$ y & y $\leqslant$ z **implies** x $\leqslant$ z.

Theorem AXIOMS:23. x $\leqslant$ y **or** y $\leqslant$ x.

Theorem AXIOMS:24. x $\leqslant$ y **implies** x+z $\leqslant$ y+z.

Theorem AXIOMS:25. x $\leqslant$ y & 0 $\leqslant$ z **implies** x·z $\leqslant$ y·z.

Theorem AXIOMS:26. **for** X, Y **being** Subset **of** REAL **st** (**ex** x **st** x $\in$ X) & (**ex** x **st** x $\in$ Y) & **for** x, y **st** x $\in$ X & y $\in$ Y **holds** x $\leqslant$ y **ex** z **st** **for** x, y **st** x $\in$ X & y $\in$ Y **holds** x $\leqslant$ z & z $\leqslant$ y.

Theorem AXIOMS:27. x **is** Real.

Theorem AXIOMS:28. x $\in$ NAT **implies** x+1 $\in$ NAT.

Theorem AXIOMS:29. **for** A **being set of** Real **st** 0 $\in$ A & **for** x **st** x $\in$ A **holds** x+1 $\in$ A **holds** NAT $\subseteq$ A.

Theorem AXIOMS:30. x $\in$ NAT **implies** x **is** Nat.

Chapter 4

BOOLE

Boolean Properties of Sets

by

Zinaida Trybulec [1]

Warsaw University (Białystok)

Halina Święczkowska [2]

Warsaw University (Białystok)

Summary. The text includes a number of theorems about Boolean operations on sets: union, intersection, difference, symmetric difference; and relations on sets: meets (having non-empty intersection), misses (being disjoint) and $\subseteq$ (inclusion).

The symbols used in this article are introduced in vocabularies FAM_OP and BOOLE. The terminology and notation used here have been introduced in article TARSKI.

reserve x, y, z **for** Any, X, Y, Z, V **for** set.

scheme Separation{A() $\to$ set, P[Any]}: **ex** X **st for** x **holds** x $\in$ X **iff** x $\in$ A() & P[x].

Definition

func $\emptyset \to$ set **means not ex** x **st** x $\in$ **it**.

let X, Y.

[1]Supported by RPBP.III-24.C1.

[2]Supported by RPBP.III-24.C1.

func $X \cup Y \to$ set **means** $x \in$ **it iff** $x \in X$ **or** $x \in Y$.

func $X \cap Y \to$ set **means** $x \in$ **it iff** $x \in X$ & $x \in Y$.

func $X \setminus Y \to$ set **means** $x \in$ **it iff** $x \in X$ & **not** $x \in Y$.

pred X meets Y **means ex** x **st** $x \in X$ & $x \in Y$.

pred X misses Y **means for** x **holds** $x \in X$ **implies not** $x \in Y$.

Definition

let X, Y.

func $X \dot{-} Y \to$ set **means it** $= (X \setminus Y) \cup (Y \setminus X)$.

Theorem BOOLE:1. $Z = \emptyset$ **iff not ex** x **st** $x \in Z$.

Theorem BOOLE:2. $Z = X \cup Y$ **iff for** x **holds** $x \in Z$ **iff** $x \in X$ **or** $x \in Y$.

Theorem BOOLE:3. $Z = X \cap Y$ **iff for** x **holds** $x \in Z$ **iff** $x \in X$ & $x \in Y$.

Theorem BOOLE:4. $Z = X \setminus Y$ **iff for** x **holds** $x \in Z$ **iff** $x \in X$ & **not** $x \in Y$.

Theorem BOOLE:5. $X \subseteq Y$ **iff for** x **holds** $x \in X$ **implies** $x \in Y$.

Theorem BOOLE:6. X meets Y **iff ex** x **st** $x \in X$ & $x \in Y$.

Theorem BOOLE:7. X misses Y **iff for** x **holds** $x \in X$ **implies not** $x \in Y$.

Definition

let X, Y.

redefine

pred $X = Y$ **means** $X \subseteq Y$ & $Y \subseteq X$.

Theorem BOOLE:8. $x \in X \cup Y$ **iff** $x \in X$ **or** $x \in Y$.

Theorem BOOLE:9. $x \in X \cap Y$ **iff** $x \in X$ & $x \in Y$.

Theorem BOOLE:10. $x \in X \setminus Y$ **iff** $x \in X$ & **not** $x \in Y$.

Theorem BOOLE:11. $x \in X$ & $X \subseteq Y$ **implies** $x \in Y$.

Theorem BOOLE:12. $x \in X$ & X misses Y **implies not** $x \in Y$.

Theorem BOOLE:13. $x \in X$ & $x \in Y$ **implies** X meets Y.

Theorem BOOLE:14. $x \in X$ **implies** $X \neq \emptyset$.

Theorem BOOLE:15. X meets Y **implies ex** x **st** $x \in X$ & $x \in Y$.

Theorem BOOLE:16. (**for** x **st** $x \in X$ **holds** $x \in Y$) **implies** $X \subseteq Y$.

Theorem BOOLE:17. (**for** x **st** $x \in X$ **holds not** $x \in Y$) **implies** X misses Y.

Theorem BOOLE:18. (**for** x **holds** $x \in X$ **iff** $x \in Y$ **or** $x \in Z$) **implies** $X = Y \cup Z$.

Theorem BOOLE:19. (**for** x **holds** $x \in X$ **iff** $x \in Y$ & $x \in Z$) **implies** $X = Y \cap Z$.

Theorem BOOLE:20. (**for** x **holds** $x \in X$ **iff** $x \in Y$ & **not** $x \in Z$) **implies** $X = Y \setminus Z$.

Theorem BOOLE:21. **not** (**ex** x **st** $x \in X$) **implies** $X = \emptyset$.

Theorem BOOLE:22. (**for** x **holds** $x \in X$ **iff** $x \in Y$) **implies** $X = Y$.

Theorem BOOLE:23. $x \in X \dot{-} Y$ **iff not** ($x \in X$ **iff** $x \in Y$).

Theorem BOOLE:24. $x \in X$ & $x \in Y$ **implies** $X \cap Y \neq \emptyset$.

Theorem BOOLE:25. (**for** x **holds not** $x \in X$ **iff** ($x \in Y$ **iff** $x \in Z$)) **implies** $X = Y \dot{-} Z$.

Theorem BOOLE:26. $X \subseteq X$.

Theorem BOOLE:27. $\emptyset \subseteq X$.

Theorem BOOLE:28. $X \subseteq Y$ & $Y \subseteq X$ **implies** $X = Y$.

Theorem BOOLE:29. $X \subseteq Y$ & $Y \subseteq Z$ **implies** $X \subseteq Z$.

Theorem BOOLE:30. $X \subseteq \emptyset$ **implies** $X = \emptyset$.

Theorem BOOLE:31. $X \subseteq X \cup Y$ & $Y \subseteq X \cup Y$.

Theorem BOOLE:32. $X \subseteq Z$ & $Y \subseteq Z$ **implies** $X \cup Y \subseteq Z$.

Theorem BOOLE:33. $X \subseteq Y$ **implies** $X \cup Z \subseteq Y \cup Z$ & $Z \cup X \subseteq Z \cup Y$.

Theorem BOOLE:34. $X \subseteq Y$ & $Z \subseteq V$ **implies** $X \cup Z \subseteq Y \cup V$.

Theorem BOOLE:35. $X \subseteq Y$ **implies** $X \cup Y = Y$ & $Y \cup X = Y$.

Theorem BOOLE:36. $X \cup Y = Y$ **or** $Y \cup X = Y$ **implies** $X \subseteq Y$.

Theorem BOOLE:37. $X \cap Y \subseteq X$ & $X \cap Y \subseteq Y$.

Theorem BOOLE:38. $X \cap Y \subseteq X \cup Z$.

Theorem BOOLE:39. $Z \subseteq X$ & $Z \subseteq Y$ **implies** $Z \subseteq X \cap Y$.

Theorem BOOLE:40. $X \subseteq Y$ **implies** $X \cap Z \subseteq Y \cap Z$ & $Z \cap X \subseteq Z \cap Y$.

Theorem BOOLE:41. $X \subseteq Y$ & $Z \subseteq V$ **implies** $X \cap Z \subseteq Y \cap V$.

Theorem BOOLE:42. $X \subseteq Y$ **implies** $X \cap Y = X$ & $Y \cap X = X$.

Theorem BOOLE:43. $X \cap Y = X$ **or** $Y \cap X = X$ **implies** $X \subseteq Y$.

Theorem BOOLE:44. $X \subseteq Z$ **implies** $X \cup Y \cap Z = (X \cup Y) \cap Z$.

Theorem BOOLE:45. $X \setminus Y = \emptyset$ **iff** $X \subseteq Y$.

Theorem BOOLE:46. $X \subseteq Y$ **implies** $X \setminus Z \subseteq Y \setminus Z$.

Theorem BOOLE:47. $X \subseteq Y$ **implies** $Z \setminus Y \subseteq Z \setminus X$.

Theorem BOOLE:48. $X \subseteq Y$ & $Z \subseteq V$ **implies** $X \setminus V \subseteq Y \setminus Z$.

Theorem BOOLE:49. $X \setminus Y \subseteq X$.

Theorem BOOLE:50. $X \subseteq Y \setminus X$ **implies** $X = \emptyset$.

Theorem BOOLE:51. $X \subseteq Y$ & $X \subseteq Z$ & $Y \cap Z = \emptyset$ **implies** $X = \emptyset$.

Theorem BOOLE:52. $X \subseteq Y \cup Z$ **implies** $X \setminus Y \subseteq Z$ & $X \setminus Z \subseteq Y$.

Theorem BOOLE:53. $(X \cap Y) \cup (X \cap Z) = X$ **implies** $X \subseteq Y \cup Z$.

Theorem BOOLE:54. $X \subseteq Y$ **implies** $Y = X \cup (Y \setminus X)$ & $Y = (Y \setminus X) \cup X$.

Theorem BOOLE:55. $X \subseteq Y$ & $Y \cap Z = \emptyset$ **implies** $X \cap Z = \emptyset$.

Theorem BOOLE:56. $X = Y \cup Z$ **iff** $Y \subseteq X$ & $Z \subseteq X$ & **for** V **st** $Y \subseteq V$ & $Z \subseteq V$ **holds** $X \subseteq V$.

Theorem BOOLE:57. $X = Y \cap Z$ **iff** $X \subseteq Y$ & $X \subseteq Z$ & **for** V **st** $V \subseteq Y$ & $V \subseteq Z$ **holds** $V \subseteq X$.

Theorem BOOLE:58. $X \setminus Y \subseteq X \dot{-} Y$.

Theorem BOOLE:59. $X \cup Y = \emptyset$ **iff** $X = \emptyset$ & $Y = \emptyset$.

Theorem BOOLE:60. $X \cup \emptyset = X$ & $\emptyset \cup X = X$.

Theorem BOOLE:61. $X \cap \emptyset = \emptyset$ & $\emptyset \cap X = \emptyset$.

Theorem BOOLE:62. $X \cup X = X$.

Theorem BOOLE:63. $X \cup Y = Y \cup X$.

Theorem BOOLE:64. $(X \cup Y) \cup Z = X \cup (Y \cup Z)$.

Theorem BOOLE:65. $X \cap X = X$.

Theorem BOOLE:66. $X \cap Y = Y \cap X$.

Theorem BOOLE:67. $(X \cap Y) \cap Z = X \cap (Y \cap Z)$.

Theorem BOOLE:68. $X \cap (X \cup Y) = X$ & $(X \cup Y) \cap X = X$ & $X \cap (Y \cup X) = X$ & $(Y \cup X) \cap X = X$.

Theorem BOOLE:69. $X \cup (X \cap Y) = X$ & $(X \cap Y) \cup X = X$ & $X \cup (Y \cap X) = X$ & $(Y \cap X) \cup X = X$.

Theorem BOOLE:70. $X \cap (Y \cup Z) = X \cap Y \cup X \cap Z$ & $(Y \cup Z) \cap X = Y \cap X \cup Z \cap X$.

Theorem BOOLE:71. $X \cup Y \cap Z = (X \cup Y) \cap (X \cup Z)$ & $Y \cap Z \cup X = (Y \cup X) \cap (Z \cup X)$.

Theorem BOOLE:72. $(X \cap Y) \cup (Y \cap Z) \cup (Z \cap X) = (X \cup Y) \cap (Y \cup Z) \cap (Z \cup X)$.

Theorem BOOLE:73. $X \setminus X = \emptyset$.

Theorem BOOLE:74. $X \setminus \emptyset = X$.

Theorem BOOLE:75. $\emptyset \setminus X = \emptyset$.

Theorem BOOLE:76. $X \setminus (X \cup Y) = \emptyset$ & $X \setminus (Y \cup X) = \emptyset$.

Theorem BOOLE:77. $X \setminus X \cap Y = X \setminus Y$ & $X \setminus Y \cap X = X \setminus Y$.

Theorem BOOLE:78. $(X \setminus Y) \cap Y = \emptyset$ & $Y \cap (X \setminus Y) = \emptyset$.

Theorem BOOLE:79. $X \cup (Y \setminus X) = X \cup Y$ & $(Y \setminus X) \cup X = Y \cup X$.

Theorem BOOLE:80. $X \cap Y \cup (X \setminus Y) = X$ & $(X \setminus Y) \cup X \cap Y = X$.

Theorem BOOLE:81. $X \setminus (Y \setminus Z) = (X \setminus Y) \cup X \cap Z$.

Theorem BOOLE:82. $X \setminus (X \setminus Y) = X \cap Y$.

Theorem BOOLE:83. $(X \cup Y) \setminus Y = X \setminus Y$.

Theorem BOOLE:84. $X \cap Y = \emptyset$ **iff** $X \setminus Y = X$.

Theorem BOOLE:85. $X \setminus (Y \cup Z) = (X \setminus Y) \cap (X \setminus Z)$.

Theorem BOOLE:86. $X \setminus (Y \cap Z) = (X \setminus Y) \cup (X \setminus Z)$.

Theorem BOOLE:87. $(X \cup Y) \setminus (X \cap Y) = (X \setminus Y) \cup (Y \setminus X)$.

Theorem BOOLE:88. $(X \setminus Y) \setminus Z = X \setminus (Y \cup Z)$.

Theorem BOOLE:89. $(X \cup Y) \setminus Z = (X \setminus Z) \cup (Y \setminus Z)$.

Theorem BOOLE:90. $X \setminus Y = Y \setminus X$ **implies** $X = Y$.

Theorem BOOLE:91. $X \dot{-} Y = (X \setminus Y) \cup (Y \setminus X)$.

Theorem BOOLE:92. $X \dot{-} \emptyset = X$ & $\emptyset \dot{-} X = X$.

Theorem BOOLE:93. $X \dot{-} X = \emptyset$.

Theorem BOOLE:94. $X \dot{-} Y = Y \dot{-} X$.

Theorem BOOLE:95. $X \cup Y = (X \dot{-} Y) \cup X \cap Y$.

Theorem BOOLE:96. $X \dot{-} Y = (X \cup Y) \setminus X \cap Y$.

Theorem BOOLE:97. $(X \dot{-} Y) \setminus Z = (X \setminus (Y \cup Z)) \cup (Y \setminus (X \cup Z))$.

Theorem BOOLE:98. $X \setminus (Y \dot{-} Z) = X \setminus (Y \cup Z) \cup X \cap Y \cap Z$.

Theorem BOOLE:99. $(X \dot{-} Y) \dot{-} Z = X \dot{-} (Y \dot{-} Z)$.

Theorem BOOLE:100. X meets $Y \cup Z$ **iff** X meets Y **or** X meets Z.

Theorem BOOLE:101. X meets Y & $Y \subseteq Z$ **implies** X meets Z.

Theorem BOOLE:102. X meets $Y \cap Z$ **implies** X meets Y & X meets Z.

Theorem BOOLE:103. X meets Y **implies** Y meets X.

Theorem BOOLE:104. **not** (X meets $\emptyset$ **or** $\emptyset$ meets X).

Theorem BOOLE:105. X misses Y **iff not** X meets Y.

Theorem BOOLE:106. X misses $Y \cup Z$ **iff** X misses Y & X misses Z.

Theorem BOOLE:107. X misses Z & $Y \subseteq Z$ **implies** X misses Y.

Theorem BOOLE:108. X misses Y **or** X misses Z **implies** X misses $Y \cap Z$.

Theorem BOOLE:109. X misses $\emptyset$ & $\emptyset$ misses X.

Theorem BOOLE:110. X meets X **iff** $X \neq \emptyset$.

Theorem BOOLE:111. $X \cap Y$ misses $X \setminus Y$.

Theorem BOOLE:112. $X \cap Y$ misses $X \dot{-} Y$.

Theorem BOOLE:113. X meets $Y \setminus Z$ **implies** X meets Y.

Theorem BOOLE:114. $X \subseteq Y$ & $X \subseteq Z$ & Y misses Z **implies** $X = \emptyset$.

Theorem BOOLE:115. $X \setminus Y \subseteq Z$ & $Y \setminus X \subseteq Z$ **implies** $X \dot{-} Y \subseteq Z$.

Theorem BOOLE:116. $X \cap (Y \setminus Z) = (X \cap Y) \setminus Z$.

Theorem BOOLE:117. $X \cap (Y \setminus Z) = X \cap Y \setminus X \cap Z$ & $(Y \setminus Z) \cap X = Y \cap X \setminus Z \cap X$.

Theorem BOOLE:118. X misses Y **iff** $X \cap Y = \emptyset$.

Theorem BOOLE:119. X meets Y **iff** $X \cap Y \neq \emptyset$.

Theorem BOOLE:120. $X \subseteq (Y \cup Z)$ & $X \cap Z = \emptyset$ **implies** $X \subseteq Y$.

Theorem BOOLE:121. $Y \subseteq X$ & $X \cap Y = \emptyset$ **implies** $Y = \emptyset$.

Theorem BOOLE:122. X misses Y **implies** Y misses X.

Chapter 5

ZFMISC_1

Some Basic Properties of Sets

by

Czesław Byliński [1]

Warsaw University (Białystok)

Summary. In this article some basic theorems about singletons, pairs, power sets, unions of families of sets, and the cartesian product of two sets are proved.

The symbols used in this article are introduced in vocabularies BOOLE and FAM_OP. The articles TARSKI and BOOLE provide the terminology and notation for this article.

Theorem ZFMISC_1:1. bool $\emptyset = \{\emptyset\}$.

Theorem ZFMISC_1:2. $\bigcup\emptyset = \emptyset$.

reserve v, x, x1, x2, y, y1, y2, z **for** Any.

reserve A, B, X, X1, X2, Y, Y1, Y2, Z **for** set.

Theorem ZFMISC_1:3. $\{x\} \neq \emptyset$.

Theorem ZFMISC_1:4. $\{x, y\} \neq \emptyset$.

Theorem ZFMISC_1:5. $\{x\} = \{x, x\}$.

Theorem ZFMISC_1:6. $\{x\} = \{y\}$ **implies** x = y.

Theorem ZFMISC_1:7. $\{x1, x2\} = \{x2, x1\}$.

Theorem ZFMISC_1:8. $\{x\} = \{y1, y2\}$ **implies** x = y1 & x = y2.

[1] Supported by RPBP.III-24.C1.

Theorem ZFMISC_1:9. $\{x\} = \{y1, y2\}$ **implies** $y1 = y2$.

Theorem ZFMISC_1:10. $\{x1, x2\} = \{y1, y2\}$ **implies** ($x1 = y1$ **or** $x1 = y2$) & ($x2 = y1$ **or** $x2 = y2$).

Theorem ZFMISC_1:11. $\{x1, x2\} = \{x1\} \cup \{x2\}$.

Theorem ZFMISC_1:12. $\{x\} \subseteq \{x, y\}$ & $\{y\} \subseteq \{x, y\}$.

Theorem ZFMISC_1:13. $\{x\} \cup \{y\} = \{x\}$ **or** $\{x\} \cup \{y\} = \{y\}$ **implies** $x = y$.

Theorem ZFMISC_1:14. $\{x\} \cup \{x, y\} = \{x, y\}$ & $\{x, y\} \cup \{x\} = \{x, y\}$.

Theorem ZFMISC_1:15. $\{y\} \cup \{x, y\} = \{x, y\}$ & $\{x, y\} \cup \{y\} = \{x, y\}$.

Theorem ZFMISC_1:16. $\{x\} \cap \{y\} = \emptyset$ **or** $\{y\} \cap \{x\} = \emptyset$ **implies** $x \neq y$.

Theorem ZFMISC_1:17. $x \neq y$ **implies** $\{x\} \cap \{y\} = \emptyset$ & $\{y\} \cap \{x\} = \emptyset$.

Theorem ZFMISC_1:18. $\{x\} \cap \{y\} = \{x\}$ **or** $\{x\} \cap \{y\} = \{y\}$ **implies** $x = y$.

Theorem ZFMISC_1:19. $\{x\} \cap \{x, y\} = \{x\}$ & $\{y\} \cap \{x, y\} = \{y\}$ & $\{x, y\} \cap \{x\} = \{x\}$ & $\{x, y\} \cap \{y\} = \{y\}$.

Theorem ZFMISC_1:20. $\{x\} \setminus \{y\} = \{x\}$ **iff** $x \neq y$.

Theorem ZFMISC_1:21. $\{x\} \setminus \{y\} = \emptyset$ **implies** $x = y$.

Theorem ZFMISC_1:22. $\{x\} \setminus \{x, y\} = \emptyset$ & $\{y\} \setminus \{x, y\} = \emptyset$.

Theorem ZFMISC_1:23. $x \neq y$ **implies** $\{x, y\} \setminus \{y\} = \{x\}$ & $\{x, y\} \setminus \{x\} = \{y\}$.

Theorem ZFMISC_1:24. $\{x\} \subseteq \{y\}$ **implies** $\{x\} = \{y\}$.

Theorem ZFMISC_1:25. $\{z\} \subseteq \{x, y\}$ **implies** $z = x$ **or** $z = y$.

Theorem ZFMISC_1:26. $\{x, y\} \subseteq \{z\}$ **implies** $x = z$ & $y = z$.

Theorem ZFMISC_1:27. $\{x, y\} \subseteq \{z\}$ **implies** $\{x, y\} = \{z\}$.

Theorem ZFMISC_1:28. $\{x1, x2\} \subseteq \{y1, y2\}$ **implies** ($x1 = y1$ **or** $x1 = y2$) & ($x2 = y1$ **or** $x2 = y2$).

Theorem ZFMISC_1:29. $x \neq y$ **implies** $\{x\} \dot{-} \{y\} = \{x, y\}$.

Theorem ZFMISC_1:30. $\mathsf{bool}\ \{x\} = \{\emptyset, \{x\}\}$.

Theorem ZFMISC_1:31. $\bigcup \{x\} = x$.

Theorem ZFMISC_1:32. $\bigcup \{\{x\}, \{y\}\} = \{x, y\}$.

Theorem ZFMISC_1:33. $[x1, x2] = [y1, y2]$ **implies** $x1 = y1$ & $x2 = y2$.

Theorem ZFMISC_1:34. $[x, y] \in [\![\{x1\}, \{y1\}]\!]$ **iff** $x = x1$ & $y = y1$.

Theorem ZFMISC_1:35. $[\![\{x\}, \{y\}]\!] = \{[x, y]\}$.

Theorem ZFMISC_1:36. $[\![\{x\}, \{y, z\}]\!] = \{[x, y], [x, z]\}$ & $[\![\{x, y\}, \{z\}]\!] = \{[x, z], [y, z]\}$.

Theorem ZFMISC_1:37. $\{x\} \subseteq X$ **iff** $x \in X$.

Theorem ZFMISC_1:38. $\{x1, x2\} \subseteq Z$ **iff** $x1 \in Z$ & $x2 \in Z$.

Theorem ZFMISC_1:39. $Y \subseteq \{x\}$ **iff** $Y = \emptyset$ **or** $Y = \{x\}$.

Theorem ZFMISC_1:40. $Y \subseteq X$ & **not** $x \in Y$ **implies** $Y \subseteq X \setminus \{x\}$.

Theorem ZFMISC_1:41. $X \neq \{x\}$ & $x \in X$ **implies** **ex** y **st** $y \in X$ & $y \neq x$.

Theorem ZFMISC_1:42. $Z \subseteq \{x1, x2\}$ **iff** $Z = \emptyset$ **or** $Z = \{x1\}$ **or** $Z = \{x2\}$ **or** $Z = \{x1, x2\}$.

Theorem ZFMISC_1:43. $\{z\} = X \cup Y$ **implies** $X = \{z\}$ & $Y = \{z\}$ **or** $X = \emptyset$ & $Y = \{z\}$ **or** $X = \{z\}$ & $Y = \emptyset$.

Theorem ZFMISC_1:44. $\{z\} = X \cup Y$ & $X \neq Y$ **implies** $X = \emptyset$ **or** $Y = \emptyset$.

Theorem ZFMISC_1:45. $\{x\} \cup X = X$ **or** $X \cup \{x\} = X$ **implies** $x \in X$.

Theorem ZFMISC_1:46. $x \in X$ **implies** $\{x\} \cup X = X$ & $X \cup \{x\} = X$.

Theorem ZFMISC_1:47. $\{x, y\} \cup Z = Z$ **or** $Z \cup \{x, y\} = Z$ **implies** $x \in Z$ & $y \in Z$.

Theorem ZFMISC_1:48. $x \in Z$ & $y \in Z$ **implies** $\{x, y\} \cup Z = Z$ & $Z \cup \{x, y\} = Z$.

Theorem ZFMISC_1:49. $\{x\} \cup X \neq \emptyset$ & $X \cup \{x\} \neq \emptyset$.

Theorem ZFMISC_1:50. $\{x, y\} \cup X \neq \emptyset$ & $X \cup \{x, y\} \neq \emptyset$.

Theorem ZFMISC_1:51. $X \cap \{x\} = \{x\}$ **or** $\{x\} \cap X = \{x\}$ **implies** $x \in X$.

Theorem ZFMISC_1:52. $x \in X$ **implies** $X \cap \{x\} = \{x\}$ & $\{x\} \cap X = \{x\}$.

Theorem ZFMISC_1:53. $x \in Z$ & $y \in Z$ **implies** $\{x, y\} \cap Z = \{x, y\}$ & $\{x, y\} = Z \cap \{x, y\}$.

Theorem ZFMISC_1:54. $\{x\} \cap X = \emptyset$ **or** $X \cap \{x\} = \emptyset$ **implies** **not** $x \in X$.

Theorem ZFMISC_1:55. $\{x, y\} \cap Z = \emptyset$ **or** $Z \cap \{x, y\} = \emptyset$ **implies** **not** $x \in Z$ & **not** $y \in Z$.

Theorem ZFMISC_1:56. **not** $x \in X$ **implies** $\{x\} \cap X = \emptyset$ & $X \cap \{x\} = \emptyset$.

Theorem ZFMISC_1:57. **not** $x \in Z$ & **not** $y \in Z$ **implies** $\{x, y\} \cap Z = \emptyset$ & $Z \cap \{x, y\} = \emptyset$.

Theorem ZFMISC_1:58. $\{x\} \cap X = \emptyset$ **or** $\{x\} \cap X = \{x\}$ & $X \cap \{x\} = \{x\}$.

Theorem ZFMISC_1:59. $\{x, y\} \cap X = \{x\}$ **or** $X \cap \{x, y\} = \{x\}$ **implies** **not** $y \in X$ **or** $x = y$.

Theorem ZFMISC_1:60. $x \in X$ & (**not** $y \in X$ **or** $x = y$) **implies** $\{x, y\} \cap X = \{x\}$ & $X \cap \{x, y\} = \{x\}$.

Theorem ZFMISC_1:61. $\{x, y\} \cap X = \{y\}$ **or** $X \cap \{x, y\} = \{y\}$ **implies** **not** $x \in X$ **or** $x = y$.

Theorem ZFMISC_1:62. $y \in X$ & (**not** $x \in X$ **or** $x = y$) **implies** $\{x, y\} \cap X = \{y\}$ & $X \cap \{x, y\} = \{y\}$.

Theorem ZFMISC_1:63. $\{x, y\} \cap X = \{x, y\}$ **or** $X \cap \{x, y\} = \{x, y\}$ **implies** $x \in X$ & $y \in X$.

Theorem ZFMISC_1:64. $z \in X \setminus \{x\}$ **iff** $z \in X$ & $z \neq x$.

Theorem ZFMISC_1:65. $X \setminus \{x\} = X$ **iff** **not** $x \in X$.

Theorem ZFMISC_1:66. $X \setminus \{x\} = \emptyset$ **implies** $X = \emptyset$ **or** $X = \{x\}$.

Theorem ZFMISC_1:67. $\{x\} \setminus X = \{x\}$ **iff** **not** $x \in X$.

Theorem ZFMISC_1:68. $\{x\} \setminus X = \emptyset$ **iff** $x \in X$.

Theorem ZFMISC_1:69. $\{x\} \setminus X = \emptyset$ **or** $\{x\} \setminus X = \{x\}$.

Theorem ZFMISC_1:70. $\{x, y\} \setminus X = \{x\}$ **iff** **not** $x \in X$ & ($y \in X$ **or** $x = y$).

Theorem ZFMISC_1:71. $\{x, y\} \setminus X = \{y\}$ **iff** ($x \in X$ **or** $x = y$) & **not** $y \in X$.

Theorem ZFMISC_1:72. $\{x, y\} \setminus X = \{x, y\}$ **iff** **not** $x \in X$ & **not** $y \in X$.

Theorem ZFMISC_1:73. $\{x, y\} \setminus X = \emptyset$ **iff** $x \in X$ & $y \in X$.

Theorem ZFMISC_1:74. $\{x, y\} \setminus X = \emptyset$ **or** $\{x, y\} \setminus X = \{x\}$ **or** $\{x, y\} \setminus X = \{y\}$ **or** $\{x, y\} \setminus X = \{x, y\}$.

Theorem ZFMISC_1:75. $X \setminus \{x, y\} = \emptyset$ **iff** $X = \emptyset$ **or** $X = \{x\}$ **or** $X = \{y\}$ **or** $X = \{x, y\}$.

Theorem ZFMISC_1:76. $\emptyset \in \mathsf{bool}\ A$.

Theorem ZFMISC_1:77. $A \in \mathsf{bool}\ A$.

Theorem ZFMISC_1:78. $\mathsf{bool}\ A \neq \emptyset$.

Theorem ZFMISC_1:79. $A \subseteq B$ **implies** $\mathsf{bool}\ A \subseteq \mathsf{bool}\ B$.

Theorem ZFMISC_1:80. $\{A\} \subseteq \mathsf{bool}\ A$.

Theorem ZFMISC_1:81. $\mathsf{bool}\ A \cup \mathsf{bool}\ B \subseteq \mathsf{bool}\ (A \cup B)$.

Theorem ZFMISC_1:82. $\mathsf{bool}\ A \cup \mathsf{bool}\ B = \mathsf{bool}\ (A \cup B)$ **implies** $A \subseteq B$ **or** $B \subseteq A$.

Theorem ZFMISC_1:83. $\mathsf{bool}\ (A \cap B) = \mathsf{bool}\ A \cap \mathsf{bool}\ B$.

Theorem ZFMISC_1:84. $\mathsf{bool}\ (A \setminus B) \subseteq \{\emptyset\} \cup (\mathsf{bool}\ A \setminus \mathsf{bool}\ B)$.

Theorem ZFMISC_1:85. $X \in \mathsf{bool}\ (A \setminus B)$ **iff** $X \subseteq A$ & X misses B.

Theorem ZFMISC_1:86. $\mathsf{bool}\ (A \setminus B) \cup \mathsf{bool}\ (B \setminus A) \subseteq \mathsf{bool}\ (A \dot{-} B)$.

Theorem ZFMISC_1:87. $X \in \mathsf{bool}\ (A \dot{-} B)$ **iff** $X \subseteq A \cup B$ & X misses $A \cap B$.

Theorem ZFMISC_1:88. $X \in \mathsf{bool}\ A$ & $Y \in \mathsf{bool}\ A$ **implies** $X \cup Y \in \mathsf{bool}\ A$.

Theorem ZFMISC_1:89. $X \in \mathsf{bool}\ A$ **or** $Y \in \mathsf{bool}\ A$ **implies** $X \cap Y \in \mathsf{bool}\ A$.

Theorem ZFMISC_1:90. $X \in \mathsf{bool}\ A$ **implies** $X \setminus Y \in \mathsf{bool}\ A$.

Theorem ZFMISC_1:91. $X \in \mathsf{bool}\ A$ & $Y \in \mathsf{bool}\ A$ **implies** $X \dot{-} Y \in \mathsf{bool}\ A$.

Theorem ZFMISC_1:92. $X \in A$ **implies** $X \subseteq \bigcup A$.

Theorem ZFMISC_1:93. $\bigcup\{X, Y\} = X \cup Y$.

Theorem ZFMISC_1:94. (**for** X **st** $X \in A$ **holds** $X \subseteq Z$) **implies** $\bigcup A \subseteq Z$.

Theorem ZFMISC_1:95. $A \subseteq B$ **implies** $\bigcup A \subseteq \bigcup B$.

Theorem ZFMISC_1:96. $\bigcup(A \cup B) = \bigcup A \cup \bigcup B$.

Theorem ZFMISC_1:97. $\bigcup(A \cap B) \subseteq \bigcup A \cap \bigcup B$.

Theorem ZFMISC_1:98. (**for** X **st** $X \in A$ **holds** $X \cap B = \emptyset$) **implies** $\bigcup(A) \cap B = \emptyset$.

Theorem ZFMISC_1:99. $\bigcup \mathsf{bool}\ A = A$.

Theorem ZFMISC_1:100. $A \subseteq \mathsf{bool} \bigcup A$.

Theorem ZFMISC_1:101. $(\mathbf{for}\ X, Y\ \mathbf{st}\ X \neq Y \ \&\ X \in A \cup B \ \&\ Y \in A \cup B\ \mathbf{holds}\ X \cap Y = \emptyset)\ \mathbf{implies} \bigcup (A \cap B) = \bigcup A \cap \bigcup B$.

Theorem ZFMISC_1:102. $z \in [\![X, Y]\!]\ \mathbf{implies\ ex}\ x, y\ \mathbf{st}\ [x, y] = z$.

Theorem ZFMISC_1:103. $A \subseteq [\![X, Y]\!] \ \&\ z \in A\ \mathbf{implies\ ex}\ x, y\ \mathbf{st}\ x \in X \ \&\ y \in Y \ \&\ z = [x, y]$.

Theorem ZFMISC_1:104. $z \in [\![X1, Y1]\!] \cap [\![X2, Y2]\!]\ \mathbf{implies\ ex}\ x, y\ \mathbf{st}\ z = [x, y] \ \&\ x \in X1 \cap X2 \ \&\ y \in Y1 \cap Y2$.

Theorem ZFMISC_1:105. $[\![X, Y]\!] \subseteq \mathsf{bool}\ \mathsf{bool}\ (X \cup Y)$.

Theorem ZFMISC_1:106. $[x, y] \in [\![X, Y]\!]\ \mathbf{iff}\ x \in X \ \&\ y \in Y$.

Theorem ZFMISC_1:107. $[x, y] \in [\![X, Y]\!]\ \mathbf{implies}\ [y, x] \in [\![Y, X]\!]$.

Theorem ZFMISC_1:108. $(\mathbf{for}\ x, y\ \mathbf{holds}\ [x, y] \in [\![X1, Y1]\!]\ \mathbf{iff}\ [x, y] \in [\![X2, Y2]\!])\ \mathbf{implies}\ [\![X1, Y1]\!] = [\![X2, Y2]\!]$.

Theorem ZFMISC_1:109. $A \subseteq [\![X, Y]\!] \ \&\ (\mathbf{for}\ x, y\ \mathbf{st}\ [x, y] \in A\ \mathbf{holds}\ [x, y] \in B)\ \mathbf{implies}\ A \subseteq B$.

Theorem ZFMISC_1:110. $A \subseteq [\![X1, Y1]\!] \ \&\ B \subseteq [\![X2, Y2]\!] \ \&\ (\mathbf{for}\ x, y\ \mathbf{holds}\ [x, y] \in A\ \mathbf{iff}\ [x, y] \in B)\ \mathbf{implies}\ A = B$.

Theorem ZFMISC_1:111. $(\mathbf{for}\ z\ \mathbf{st}\ z \in A\ \mathbf{ex}\ x, y\ \mathbf{st}\ z = [x, y]) \ \&\ (\mathbf{for}\ x, y\ \mathbf{st}\ [x, y] \in A\ \mathbf{holds}\ [x, y] \in B)\ \mathbf{implies}\ A \subseteq B$.

Theorem ZFMISC_1:112. $(\mathbf{for}\ z\ \mathbf{st}\ z \in A\ \mathbf{ex}\ x, y\ \mathbf{st}\ z = [x, y]) \ \&\ (\mathbf{for}\ z\ \mathbf{st}\ z \in B\ \mathbf{ex}\ x, y\ \mathbf{st}\ z = [x, y]) \ \&\ (\mathbf{for}\ x, y\ \mathbf{holds}\ [x, y] \in A\ \mathbf{iff}\ [x, y] \in B)\ \mathbf{implies}\ A = B$.

Theorem ZFMISC_1:113. $[\![X, Y]\!] = \emptyset\ \mathbf{iff}\ X = \emptyset\ \mathbf{or}\ Y = \emptyset$.

Theorem ZFMISC_1:114. $X \neq \emptyset \ \&\ Y \neq \emptyset \ \&\ [\![X, Y]\!] = [\![Y, X]\!]\ \mathbf{implies}\ X = Y$.

Theorem ZFMISC_1:115. $[\![X, X]\!] = [\![Y, Y]\!]\ \mathbf{implies}\ X = Y$.

Theorem ZFMISC_1:116. $X \subseteq [\![X, X]\!]\ \mathbf{implies}\ X = \emptyset$.

Theorem ZFMISC_1:117. $Z \neq \emptyset \ \&\ ([\![X, Z]\!] \subseteq [\![Y, Z]\!]\ \mathbf{or}\ [\![Z, X]\!] \subseteq [\![Z, Y]\!])\ \mathbf{implies}\ X \subseteq Y$.

Theorem ZFMISC_1:118. $X \subseteq Y\ \mathbf{implies}\ [\![X, Z]\!] \subseteq [\![Y, Z]\!] \ \&\ [\![Z, X]\!] \subseteq [\![Z, Y]\!]$.

Theorem ZFMISC_1:119. $X1 \subseteq Y1 \ \&\ X2 \subseteq Y2\ \mathbf{implies}\ [\![X1, X2]\!] \subseteq [\![Y1, Y2]\!]$.

Theorem ZFMISC_1:120. $[\![X \cup Y, Z]\!] = [\![X, Z]\!] \cup [\![Y, Z]\!] \ \&\ [\![Z, X \cup Y]\!] = [\![Z, X]\!] \cup [\![Z, Y]\!]$.

Theorem ZFMISC_1:121. $[\![X1 \cup X2, Y1 \cup Y2]\!] = [\![X1, Y1]\!] \cup [\![X1, Y2]\!] \cup [\![X2, Y1]\!] \cup [\![X2, Y2]\!]$.

Theorem ZFMISC_1:122. $[\![X \cap Y, Z]\!] = [\![X, Z]\!] \cap [\![Y, Z]\!] \ \&\ [\![Z, X \cap Y]\!] = [\![Z, X]\!] \cap [\![Z, Y]\!]$.

Theorem ZFMISC_1:123. $[\![X1 \cap X2, Y1 \cap Y2]\!] = [\![X1, Y1]\!] \cap [\![X2, Y2]\!]$.

Theorem ZFMISC_1:124. $A \subseteq X \ \&\ B \subseteq Y\ \mathbf{implies}\ [\![A, Y]\!] \cap [\![X, B]\!] = [\![A, B]\!]$.

Theorem ZFMISC_1:125. $[\![X \setminus Y, Z]\!] = [\![X, Z]\!] \setminus [\![Y, Z]\!] \ \&\ [\![Z, X \setminus Y]\!] = [\![Z, X]\!] \setminus [\![Z, Y]\!]$.

Theorem ZFMISC_1:126. $[\![X1, X2]\!] \setminus [\![Y1, Y2]\!] = [\![X1 \setminus Y1, X2]\!] \cup [\![X1, X2 \setminus Y2]\!]$.

Theorem ZFMISC_1:127. $X1 \cap X2 = \emptyset$ **or** $Y1 \cap Y2 = \emptyset$ **implies** $\llbracket X1, Y1 \rrbracket \cap \llbracket X2, Y2 \rrbracket = \emptyset$.

Theorem ZFMISC_1:128. $[x, y] \in \llbracket \{z\}, Y \rrbracket$ **iff** $x = z$ & $y \in Y$.

Theorem ZFMISC_1:129. $[x, y] \in \llbracket X, \{z\} \rrbracket$ **iff** $x \in X$ & $y = z$.

Theorem ZFMISC_1:130. $X \neq \emptyset$ **implies** $\llbracket \{x\}, X \rrbracket \neq \emptyset$ & $\llbracket X, \{x\} \rrbracket \neq \emptyset$.

Theorem ZFMISC_1:131. $x \neq y$ **implies** $\llbracket \{x\}, X \rrbracket \cap \llbracket \{y\}, Y \rrbracket = \emptyset$ & $\llbracket X, \{x\} \rrbracket \cap \llbracket Y, \{y\} \rrbracket = \emptyset$.

Theorem ZFMISC_1:132. $\llbracket \{x, y\}, X \rrbracket = \llbracket \{x\}, X \rrbracket \cup \llbracket \{y\}, X \rrbracket$ & $\llbracket X, \{x, y\} \rrbracket = \llbracket X, \{x\} \rrbracket \cup \llbracket X, \{y\} \rrbracket$.

Theorem ZFMISC_1:133. $Z = \llbracket X, Y \rrbracket$ **iff for** z **holds** $z \in Z$ **iff ex** x, y **st** $x \in X$ & $y \in Y$ & $z = [x, y]$.

Theorem ZFMISC_1:134. $X1 \neq \emptyset$ & $Y1 \neq \emptyset$ & $\llbracket X1, Y1 \rrbracket = \llbracket X2, Y2 \rrbracket$ **implies** $X1 = X2$ & $Y1 = Y2$.

Theorem ZFMISC_1:135. $X \subseteq \llbracket X, Y \rrbracket$ **or** $X \subseteq \llbracket Y, X \rrbracket$ **implies** $X = \emptyset$.

Chapter 6

ENUMSET1

Enumerated Sets

by

Andrzej Trybulec [1]

Warsaw University (Białystok)

Summary. We prove basic facts about enumerated sets: definitional theorems and their immediate consequences, some theorems related to the decomposition of an enumerated set into union of two sets, facts about removing elements that occur more than once, and facts about permutations of enumerated sets (with the length ≤ 4). The article includes also schemes enabling instantiation of up to nine universal quantifiers.

The symbols used in this article are introduced in vocabularies BOOLE and FAM_OP. The articles TARSKI and BOOLE provide the terminology and notation for this article.

reserve x, x1, x2, x3, x4, x5, x6, x7, x8, y, y1, y2, y3, y4, y5, y6, y7, y8, z, z1, z2, z3, z4, z5, z6, z7, z8 **for** Any.

reserve X, X1, X2, Y, Y1, Y2, Z, Z1, Z2 **for** set.

scheme UI1{x1() → Any, P[Any]}: P[x1()] **provided** A: **for** x1 **holds** P[x1].

scheme UI2{x1() → Any, x2() → Any, P[Any, Any]}: P[x1(), x2()] **provided** A: **for** x1, x2 **holds** P[x1, x2].

scheme UI3{x1() → Any, x2() → Any, x3() → Any, P[Any, Any, Any]}: P[x1(), x2(), x3()] **provided** A: **for** x1, x2, x3 **holds** P[x1, x2, x3].

[1] Supported by RPBP.III-24.C1.

scheme UI4{x1() → Any, x2() → Any, x3() → Any, x4() → Any, P[Any, Any, Any, Any]}: P[x1(), x2(), x3(), x4()] **provided** A: **for** x1, x2, x3, x4 **holds** P[x1, x2, x3, x4].

scheme UI5{x1() → Any, x2() → Any, x3() → Any, x4() → Any, x5() → Any, P[Any, Any, Any, Any, Any]}: P[x1(), x2(), x3(), x4(), x5()] **provided** A: **for** x1, x2, x3, x4, x5 **holds** P[x1, x2, x3, x4, x5].

scheme UI6{x1() → Any, x2() → Any, x3() → Any, x4() → Any, x5() → Any, x6() → Any, P[Any, Any, Any, Any, Any, Any]}: P[x1(), x2(), x3(), x4(), x5(), x6()] **provided** A: **for** x1, x2, x3, x4, x5, x6 **holds** P[x1, x2, x3, x4, x5, x6].

scheme UI7{x1() → Any, x2() → Any, x3() → Any, x4() → Any, x5() → Any, x6() → Any, x7() → Any, P[Any, Any, Any, Any, Any, Any, Any]}: P[x1(), x2(), x3(), x4(), x5(), x6(), x7()] **provided** A: **for** x1, x2, x3, x4, x5, x6, x7 **holds** P[x1, x2, x3, x4, x5, x6, x7].

scheme UI8{x1() → Any, x2() → Any, x3() → Any, x4() → Any, x5() → Any, x6() → Any, x7() → Any, x8() → Any, P[Any, Any, Any, Any, Any, Any, Any, Any]}: P[x1(), x2(), x3(), x4(), x5(), x6(), x7(), x8()] **provided** A: **for** x1, x2, x3, x4, x5, x6, x7, x8 **holds** P[x1, x2, x3, x4, x5, x6, x7, x8].

scheme UI9{x1() → Any, x2() → Any, x3() → Any, x4() → Any, x5() → Any, x6() → Any, x7() → Any, x8() → Any, x9() → Any, P[Any, Any, Any, Any, Any, Any, Any, Any, Any]}: P[x1(), x2(), x3(), x4(), x5(), x6(), x7(), x8(), x9()] **provided** A: **for** x1, x2, x3, x4, x5, x6, x7, x8, x9 **being** Any **holds** P[x1, x2, x3, x4, x5, x6, x7, x8, x9].

Theorem ENUMSET1:1. **for** x1, X **holds** X = {x1} **iff** **for** x **holds** x ∈ X **iff** x = x1.

Theorem ENUMSET1:2. **for** x1, x **holds** x ∈ {x1} **iff** x = x1.

Theorem ENUMSET1:3. x ∈ {x1} **implies** x = x1.

Theorem ENUMSET1:4. x ∈ {x}.

Theorem ENUMSET1:5. **for** x1, X **st** **for** x **holds** x ∈ X **iff** x = x1 **holds** X = {x1}.

Theorem ENUMSET1:6. **for** x1, x2, X **holds** X = {x1, x2} **iff** **for** x **holds** x ∈ X **iff** x = x1 **or** x = x2.

Theorem ENUMSET1:7. **for** x1, x2 **holds** **for** x **holds** x ∈ {x1, x2} **iff** x = x1 **or** x = x2.

Theorem ENUMSET1:8. x ∈ {x1, x2} **implies** x = x1 **or** x = x2.

Theorem ENUMSET1:9. x = x1 **or** x = x2 **implies** x ∈ {x1, x2}.

Theorem ENUMSET1:10. **for** x1, x2, X **st** **for** x **holds** x ∈ X **iff** x = x1 **or** x = x2 **holds** X = {x1, x2}.

Definition

let x1, x2, x3.

func {x1, x2, x3} → set **means** x ∈ **it** **iff** x = x1 **or** x = x2 **or** x = x3.

Theorem ENUMSET1:11. **for** x1, x2, x3, X **holds** X = {x1, x2, x3} **iff** **for** x **holds** x ∈ X **iff** x = x1 **or** x = x2 **or** x = x3.

Theorem ENUMSET1:12. **for** x1, x2, x3 **holds** **for** x **holds** x ∈ {x1, x2, x3} **iff** x = x1 **or** x = x2 **or** x = x3.

Theorem ENUMSET1:13. x ∈ {x1, x2, x3} **implies** x = x1 **or** x = x2 **or** x = x3.

Theorem ENUMSET1:14. x = x1 **or** x = x2 **or** x = x3 **implies** x ∈ {x1, x2, x3}.

Theorem ENUMSET1:15. **for** x1, x2, x3, X **st** **for** x **holds** x ∈ X **iff** x = x1 **or** x = x2 **or** x = x3 **holds** X = {x1, x2, x3}.

Definition

let x1, x2, x3, x4.

func {x1, x2, x3, x4} → set **means** x ∈ **it** **iff** x = x1 **or** x = x2 **or** x = x3 **or** x = x4.

Theorem ENUMSET1:16. **for** x1, x2, x3, x4, X **holds** X = {x1, x2, x3, x4} **iff** **for** x **holds** x ∈ X **iff** x = x1 **or** x = x2 **or** x = x3 **or** x = x4.

Theorem ENUMSET1:17. **for** x1, x2, x3, x4 **holds** **for** x **holds** x ∈ {x1, x2, x3, x4} **iff** x = x1 **or** x = x2 **or** x = x3 **or** x = x4.

Theorem ENUMSET1:18. x ∈ {x1, x2, x3, x4} **implies** x = x1 **or** x = x2 **or** x = x3 **or** x = x4.

Theorem ENUMSET1:19. x = x1 **or** x = x2 **or** x = x3 **or** x = x4 **implies** x ∈ {x1, x2, x3, x4}.

Theorem ENUMSET1:20. **for** x1, x2, x3, x4, X **st** **for** x **holds** x ∈ X **iff** x = x1 **or** x = x2 **or** x = x3 **or** x = x4 **holds** X = {x1, x2, x3, x4}.

Definition

let x1, x2, x3, x4, x5.

func {x1, x2, x3, x4, x5} → set **means** x ∈ **it** **iff** x = x1 **or** x = x2 **or** x = x3 **or** x = x4 **or** x = x5.

Theorem ENUMSET1:21. **for** x1, x2, x3, x4, x5 **for** X **being** set **holds** X = {x1, x2, x3, x4, x5} **iff** **for** x **holds** x ∈ X **iff** x = x1 **or** x = x2 **or** x = x3 **or** x = x4 **or** x = x5.

Theorem ENUMSET1:22. x ∈ {x1, x2, x3, x4, x5} **iff** x = x1 **or** x = x2 **or** x = x3 **or** x = x4 **or** x = x5.

Theorem ENUMSET1:23. x ∈ {x1, x2, x3, x4, x5} **implies** x = x1 **or** x = x2 **or** x = x3 **or** x = x4 **or** x = x5.

Theorem ENUMSET1:24. x = x1 **or** x = x2 **or** x = x3 **or** x = x4 **or** x = x5 **implies** x ∈ {x1, x2, x3, x4, x5}.

Theorem ENUMSET1:25. **for** X **being** set **st** **for** x **holds** x ∈ X **iff** x = x1 **or** x = x2 **or** x = x3 **or** x = x4 **or** x = x5 **holds** X = {x1, x2, x3, x4, x5}.

Definition

let x1, x2, x3, x4, x5, x6.

func {x1, x2, x3, x4, x5, x6} → set **means** x ∈ **it** **iff** x = x1 **or** x = x2 **or** x = x3 **or** x = x4 **or** x = x5 **or** x = x6.

Theorem ENUMSET1:26. **for** x1, x2, x3, x4, x5, x6 **for** X **being** set **holds** X = {x1, x2, x3, x4, x5, x6} **iff for** x **holds** x ∈ X **iff** x = x1 **or** x = x2 **or** x = x3 **or** x = x4 **or** x = x5 **or** x = x6.

Theorem ENUMSET1:27. x ∈ {x1, x2, x3, x4, x5, x6} **iff** x = x1 **or** x = x2 **or** x = x3 **or** x = x4 **or** x = x5 **or** x = x6.

Theorem ENUMSET1:28. x ∈ {x1, x2, x3, x4, x5, x6} **implies** x = x1 **or** x = x2 **or** x = x3 **or** x = x4 **or** x = x5 **or** x = x6.

Theorem ENUMSET1:29. x = x1 **or** x = x2 **or** x = x3 **or** x = x4 **or** x = x5 **or** x = x6 **implies** x ∈ {x1, x2, x3, x4, x5, x6}.

Theorem ENUMSET1:30. **for** X **being** set **st for** x **holds** x ∈ X **iff** x = x1 **or** x = x2 **or** x = x3 **or** x = x4 **or** x = x5 **or** x = x6 **holds** X = {x1, x2, x3, x4, x5, x6}.

Definition

let x1, x2, x3, x4, x5, x6, x7.

func {x1, x2, x3, x4, x5, x6, x7} → set **means** x ∈ **it iff** x = x1 **or** x = x2 **or** x = x3 **or** x = x4 **or** x = x5 **or** x = x6 **or** x = x7.

Theorem ENUMSET1:31. **for** x1, x2, x3, x4, x5, x6, x7 **for** X **being** set **holds** X = {x1, x2, x3, x4, x5, x6, x7} **iff for** x **holds** x ∈ X **iff** x = x1 **or** x = x2 **or** x = x3 **or** x = x4 **or** x = x5 **or** x = x6 **or** x = x7.

Theorem ENUMSET1:32. x ∈ {x1, x2, x3, x4, x5, x6, x7} **iff** x = x1 **or** x = x2 **or** x = x3 **or** x = x4 **or** x = x5 **or** x = x6 **or** x = x7.

Theorem ENUMSET1:33. x ∈ {x1, x2, x3, x4, x5, x6, x7} **implies** x = x1 **or** x = x2 **or** x = x3 **or** x = x4 **or** x = x5 **or** x = x6 **or** x = x7.

Theorem ENUMSET1:34. x = x1 **or** x = x2 **or** x = x3 **or** x = x4 **or** x = x5 **or** x = x6 **or** x = x7 **implies** x ∈ {x1, x2, x3, x4, x5, x6, x7}.

Theorem ENUMSET1:35. **for** X **being** set **st for** x **holds** x ∈ X **iff** x = x1 **or** x = x2 **or** x = x3 **or** x = x4 **or** x = x5 **or** x = x6 **or** x = x7 **holds** X = {x1, x2, x3, x4, x5, x6, x7}.

Definition

let x1, x2, x3, x4, x5, x6, x7, x8.

func {x1, x2, x3, x4, x5, x6, x7, x8} → set **means** x ∈ **it iff** x = x1 **or** x = x2 **or** x = x3 **or** x = x4 **or** x = x5 **or** x = x6 **or** x = x7 **or** x = x8.

Theorem ENUMSET1:36. **for** x1, x2, x3, x4, x5, x6, x7, x8 **for** X **being** set **holds** X = {x1, x2, x3, x4, x5, x6, x7, x8} **iff for** x **holds** x ∈ X **iff** x = x1 **or** x = x2 **or** x = x3 **or** x = x4 **or** x = x5 **or** x = x6 **or** x = x7 **or** x = x8.

Theorem ENUMSET1:37. x ∈ {x1, x2, x3, x4, x5, x6, x7, x8} **iff** x = x1 **or** x = x2 **or** x = x3 **or** x = x4 **or** x = x5 **or** x = x6 **or** x = x7 **or** x = x8.

Theorem ENUMSET1:38. x ∈ {x1, x2, x3, x4, x5, x6, x7, x8} **implies** x = x1 **or** x = x2 **or** x = x3 **or** x = x4 **or** x = x5 **or** x = x6 **or** x = x7 **or** x = x8.

Theorem ENUMSET1:39. x = x1 **or** x = x2 **or** x = x3 **or** x = x4 **or** x = x5 **or** x = x6 **or** x = x7 **or** x = x8 **implies** x ∈ {x1, x2, x3, x4, x5, x6, x7, x8}.

Theorem ENUMSET1:40. **for** X **being** set **st for** x **holds** x ∈ X **iff** x = x1 **or** x = x2 **or** x = x3 **or** x = x4 **or** x = x5 **or** x = x6 **or** x = x7 **or** x = x8 **holds** X = {x1, x2, x3, x4, x5, x6, x7, x8}.

Theorem ENUMSET1:41. {x1, x2} = {x1}∪{x2}.

Theorem ENUMSET1:42. {x1, x2, x3} = {x1}∪{x2, x3}.

Theorem ENUMSET1:43. {x1, x2, x3} = {x1, x2}∪{x3}.

Theorem ENUMSET1:44. {x1, x2, x3, x4} = {x1}∪{x2, x3, x4}.

Theorem ENUMSET1:45. {x1, x2, x3, x4} = {x1, x2}∪{x3, x4}.

Theorem ENUMSET1:46. {x1, x2, x3, x4} = {x1, x2, x3}∪{x4}.

Theorem ENUMSET1:47. {x1, x2, x3, x4, x5} = {x1}∪{x2, x3, x4, x5}.

Theorem ENUMSET1:48. {x1, x2, x3, x4, x5} = {x1, x2}∪{x3, x4, x5}.

Theorem ENUMSET1:49. {x1, x2, x3, x4, x5} = {x1, x2, x3}∪{x4, x5}.

Theorem ENUMSET1:50. {x1, x2, x3, x4, x5} = {x1, x2, x3, x4}∪{x5}.

Theorem ENUMSET1:51. {x1, x2, x3, x4, x5, x6} = {x1}∪{x2, x3, x4, x5, x6}.

Theorem ENUMSET1:52. {x1, x2, x3, x4, x5, x6} = {x1, x2}∪{x3, x4, x5, x6}.

Theorem ENUMSET1:53. {x1, x2, x3, x4, x5, x6} = {x1, x2, x3}∪{x4, x5, x6}.

Theorem ENUMSET1:54. {x1, x2, x3, x4, x5, x6} = {x1, x2, x3, x4}∪{x5, x6}.

Theorem ENUMSET1:55. {x1, x2, x3, x4, x5, x6} = {x1, x2, x3, x4, x5}∪{x6}.

Theorem ENUMSET1:56. {x1, x2, x3, x4, x5, x6, x7} = {x1}∪{x2, x3, x4, x5, x6, x7}.

Theorem ENUMSET1:57. {x1, x2, x3, x4, x5, x6, x7} = {x1, x2}∪{x3, x4, x5, x6, x7}.

Theorem ENUMSET1:58. {x1, x2, x3, x4, x5, x6, x7} = {x1, x2, x3}∪{x4, x5, x6, x7}.

Theorem ENUMSET1:59. {x1, x2, x3, x4, x5, x6, x7} = {x1, x2, x3, x4}∪{x5, x6, x7}.

Theorem ENUMSET1:60. {x1, x2, x3, x4, x5, x6, x7} = {x1, x2, x3, x4, x5}∪{x6, x7}.

Theorem ENUMSET1:61. {x1, x2, x3, x4, x5, x6, x7} = {x1, x2, x3, x4, x5, x6}∪{x7}.

Theorem ENUMSET1:62. {x1, x2, x3, x4, x5, x6, x7, x8} = {x1}∪{x2, x3, x4, x5, x6, x7, x8}.

Theorem ENUMSET1:63. {x1, x2, x3, x4, x5, x6, x7, x8} = {x1, x2}∪{x3, x4, x5, x6, x7, x8}.

Theorem ENUMSET1:64. {x1, x2, x3, x4, x5, x6, x7, x8} = {x1, x2, x3}∪{x4, x5, x6, x7, x8}.

Theorem ENUMSET1:65. {x1, x2, x3, x4, x5, x6, x7, x8} = {x1, x2, x3, x4}∪{x5, x6, x7, x8}.

Theorem ENUMSET1:66. {x1, x2, x3, x4, x5, x6, x7, x8} = {x1, x2, x3, x4, x5}∪{x6, x7, x8}.

Theorem ENUMSET1:67. {x1, x2, x3, x4, x5, x6, x7, x8} = {x1, x2, x3, x4, x5, x6}∪ {x7, x8}.

Theorem ENUMSET1:68. {x1, x2, x3, x4, x5, x6, x7, x8} = {x1, x2, x3, x4, x5, x6, x7}∪{x8}.

Theorem ENUMSET1:69. {x1, x1} = {x1}.

Theorem ENUMSET1:70. {x1, x1, x2} = {x1, x2}.

Theorem ENUMSET1:71. {x1, x1, x2, x3} = {x1, x2, x3}.

Theorem ENUMSET1:72. {x1, x1, x2, x3, x4} = {x1, x2, x3, x4}.

Theorem ENUMSET1:73. {x1, x1, x2, x3, x4, x5} = {x1, x2, x3, x4, x5}.

Theorem ENUMSET1:74. {x1, x1, x2, x3, x4, x5, x6} = {x1, x2, x3, x4, x5, x6}.

Theorem ENUMSET1:75. {x1, x1, x2, x3, x4, x5, x6, x7} = {x1, x2, x3, x4, x5, x6, x7}.

Theorem ENUMSET1:76. {x1, x1, x1} = {x1}.

Theorem ENUMSET1:77. {x1, x1, x1, x2} = {x1, x2}.

Theorem ENUMSET1:78. {x1, x1, x1, x2, x3} = {x1, x2, x3}.

Theorem ENUMSET1:79. {x1, x1, x1, x2, x3, x4} = {x1, x2, x3, x4}.

Theorem ENUMSET1:80. {x1, x1, x1, x2, x3, x4, x5} = {x1, x2, x3, x4, x5}.

Theorem ENUMSET1:81. {x1, x1, x1, x2, x3, x4, x5, x6} = {x1, x2, x3, x4, x5, x6}.

Theorem ENUMSET1:82. {x1, x1, x1, x1} = {x1}.

Theorem ENUMSET1:83. {x1, x1, x1, x1, x2} = {x1, x2}.

Theorem ENUMSET1:84. {x1, x1, x1, x1, x2, x3} = {x1, x2, x3}.

Theorem ENUMSET1:85. {x1, x1, x1, x1, x2, x3, x4} = {x1, x2, x3, x4}.

Theorem ENUMSET1:86. {x1, x1, x1, x1, x2, x3, x4, x5} = {x1, x2, x3, x4, x5}.

Theorem ENUMSET1:87. {x1, x1, x1, x1, x1} = {x1}.

Theorem ENUMSET1:88. {x1, x1, x1, x1, x1, x2} = {x1, x2}.

Theorem ENUMSET1:89. {x1, x1, x1, x1, x1, x2, x3} = {x1, x2, x3}.

Theorem ENUMSET1:90. {x1, x1, x1, x1, x1, x2, x3, x4} = {x1, x2, x3, x4}.

Theorem ENUMSET1:91. {x1, x1, x1, x1, x1, x1} = {x1}.

Theorem ENUMSET1:92. {x1, x1, x1, x1, x1, x1, x2} = {x1, x2}.

Theorem ENUMSET1:93. {x1, x1, x1, x1, x1, x1, x2, x3} = {x1, x2, x3}.

Theorem ENUMSET1:94. {x1, x1, x1, x1, x1, x1, x1} = {x1}.

Theorem ENUMSET1:95. {x1, x1, x1, x1, x1, x1, x1, x2} = {x1, x2}.

Theorem ENUMSET1:96. {x1, x1, x1, x1, x1, x1, x1, x1} = {x1}.

Theorem ENUMSET1:97. {x1, x2} = {x2, x1}.

Theorem ENUMSET1:98. {x1, x2, x3} = {x1, x3, x2}.

Theorem ENUMSET1:99. {x1, x2, x3} = {x2, x1, x3}.

Theorem ENUMSET1:100. {x1, x2, x3} = {x2, x3, x1}.

Theorem ENUMSET1:101. {x1, x2, x3} = {x3, x1, x2}.

Theorem ENUMSET1:102. {x1, x2, x3} = {x3, x2, x1}.

Theorem ENUMSET1:103. {x1, x2, x3, x4} = {x1, x2, x4, x3}.

Theorem ENUMSET1:104. {x1, x2, x3, x4} = {x1, x3, x2, x4}.

Theorem ENUMSET1:105. {x1, x2, x3, x4} = {x1, x3, x4, x2}.

Theorem ENUMSET1:106. {x1, x2, x3, x4} = {x1, x4, x2, x3}.

Theorem ENUMSET1:107. {x1, x2, x3, x4} = {x1, x4, x3, x2}.

Theorem ENUMSET1:108. {x1, x2, x3, x4} = {x2, x1, x3, x4}.

Theorem ENUMSET1:109. {x1, x2, x3, x4} = {x2, x1, x4, x3}.

Theorem ENUMSET1:110. {x1, x2, x3, x4} = {x2, x3, x1, x4}.

Theorem ENUMSET1:111. {x1, x2, x3, x4} = {x2, x3, x4, x1}.

Theorem ENUMSET1:112. {x1, x2, x3, x4} = {x2, x4, x1, x3}.

Theorem ENUMSET1:113. {x1, x2, x3, x4} = {x2, x4, x3, x1}.

Theorem ENUMSET1:114. {x1, x2, x3, x4} = {x3, x1, x2, x4}.

Theorem ENUMSET1:115. {x1, x2, x3, x4} = {x3, x1, x4, x2}.

Theorem ENUMSET1:116. {x1, x2, x3, x4} = {x3, x2, x1, x4}.

Theorem ENUMSET1:117. {x1, x2, x3, x4} = {x3, x2, x4, x1}.

Theorem ENUMSET1:118. {x1, x2, x3, x4} = {x3, x4, x1, x2}.

Theorem ENUMSET1:119. {x1, x2, x3, x4} = {x3, x4, x2, x1}.

Theorem ENUMSET1:120. {x1, x2, x3, x4} = {x4, x1, x2, x3}.

Theorem ENUMSET1:121. {x1, x2, x3, x4} = {x4, x1, x3, x2}.

Theorem ENUMSET1:122. {x1, x2, x3, x4} = {x4, x2, x1, x3}.

Theorem ENUMSET1:123. {x1, x2, x3, x4} = {x4, x2, x3, x1}.

Theorem ENUMSET1:124. {x1, x2, x3, x4} = {x4, x3, x1, x2}.

Theorem ENUMSET1:125. {x1, x2, x3, x4} = {x4, x3, x2, x1}.

Chapter 7

SUBSET_1

Properties of Subsets

by

Zinaida Trybulec [1]

Warsaw University (Białystok)

Summary. The text includes theorems concerning properties of subsets, and some operations on sets. The functions yielding improper subsets of a set, i.e. the empty set and the set itself are introduced. Functions and predicates introduced for sets are redefined. Some theorems about enumerated sets are proved.

The symbols used in this article are introduced in vocabularies BOOLE and SUB_OP. The terminology and notation used in this article have been introduced in the following articles: TARSKI, BOOLE, and ENUMSET1.

reserve E, X **for** set.

reserve x, y **for** Any.

Theorem SUBSET_1:1. $E \neq \emptyset$ **implies** (x **is** Element **of** E **iff** $x \in E$).

Theorem SUBSET_1:2. $x \in E$ **implies** x **is** Element **of** E.

Theorem SUBSET_1:3. X **is** Subset **of** E **iff** $X \subseteq E$.

Definition

let E.

func $\emptyset$ E $\rightarrow$ Subset **of** E **means it** $= \emptyset$.

[1]Supported by RPBP.III-24.C1.

func ΩE $\rightarrow$ Subset **of** E **means** **it** = E.

Theorem SUBSET_1:4. $\emptyset$ **is** Subset **of** X.

Theorem SUBSET_1:5. X **is** Subset **of** X.

reserve A, B, C **for** Subset **of** E.

Theorem SUBSET_1:6. x $\in$ A **implies** x **is** Element **of** E.

Theorem SUBSET_1:7. (**for** x **being** Element **of** E **holds** x $\in$ A **implies** x $\in$ B) **implies** A $\subseteq$ B.

Theorem SUBSET_1:8. (**for** x **being** Element **of** E **holds** x $\in$ A **iff** x $\in$ B) **implies** A = B.

Theorem SUBSET_1:9. x $\in$ A **implies** x $\in$ E.

Theorem SUBSET_1:10. A $\neq$ $\emptyset$ **iff** **ex** x **being** Element **of** E **st** x $\in$ A.

Definition

let E.

let A.

func A^c $\rightarrow$ Subset **of** E **means** **it** = E$\setminus$A.

let B.

redefine

func A$\cup$B $\rightarrow$ Subset **of** E.

func A$\cap$B $\rightarrow$ Subset **of** E.

func A$\setminus$B $\rightarrow$ Subset **of** E.

func A$\dot{-}$B $\rightarrow$ Subset **of** E.

Theorem SUBSET_1:11. x $\in$ A$\cap$B **implies** x **is** Element **of** A & x **is** Element **of** B.

Theorem SUBSET_1:12. x $\in$ A$\cup$B **implies** x **is** Element **of** A **or** x **is** Element **of** B.

Theorem SUBSET_1:13. x $\in$ A$\setminus$B **implies** x **is** Element **of** A.

Theorem SUBSET_1:14. x $\in$ A$\dot{-}$B **implies** x **is** Element **of** A **or** x **is** Element **of** B.

Theorem SUBSET_1:15. (**for** x **being** Element **of** E **holds** x $\in$ A **iff** x $\in$ B **or** x $\in$ C) **implies** A = B$\cup$C.

Theorem SUBSET_1:16. (**for** x **being** Element **of** E **holds** x $\in$ A **iff** x $\in$ B & x $\in$ C) **implies** A = B$\cap$C.

Theorem SUBSET_1:17. (**for** x **being** Element **of** E **holds** x $\in$ A **iff** x $\in$ B & **not** x $\in$ C) **implies** A = B$\setminus$C.

Theorem SUBSET_1:18. (**for** x **being** Element **of** E **holds** x $\in$ A **iff** **not** (x $\in$ B **iff** x $\in$ C)) **implies** A = B$\dot{-}$C.

Theorem SUBSET_1:19. $\emptyset$ E = $\emptyset$.

Theorem SUBSET_1:20. ΩE = E.

Theorem SUBSET_1:21. $\emptyset$ E = $(\Omega E)^c$.

Theorem SUBSET_1:22. ΩE = ($\emptyset$ E)^c.

Theorem SUBSET_1:23. A^c = E$\setminus$A.

Theorem SUBSET_1:24. A^{cc} = A.

Theorem SUBSET_1:25. A$\cup$A^c = ΩE & A^c$\cup$A = ΩE.

Theorem SUBSET_1:26. A$\cap$A^c = $\emptyset$ E & A^c$\cap$A = $\emptyset$ E.

Theorem SUBSET_1:27. A$\cap$$\emptyset$ E = $\emptyset$ E & $\emptyset$ E$\cap$A = $\emptyset$ E.

Theorem SUBSET_1:28. A$\cup$$\Omega$E = ΩE & ΩE$\cup$A = ΩE.

Theorem SUBSET_1:29. (A$\cup$B)^c = A^c$\cap$B^c.

Theorem SUBSET_1:30. (A$\cap$B)^c = A^c$\cup$B^c.

Theorem SUBSET_1:31. A $\subseteq$ B **iff** B^c $\subseteq$ A^c.

Theorem SUBSET_1:32. A$\setminus$B = A$\cap$B^c.

Theorem SUBSET_1:33. (A$\setminus$B)^c = A^c$\cup$B.

Theorem SUBSET_1:34. (A$\dot{-}$B)^c = A$\cap$B$\cup$A^c$\cap$B^c.

Theorem SUBSET_1:35. A $\subseteq$ B^c **implies** B $\subseteq$ A^c.

Theorem SUBSET_1:36. A^c $\subseteq$ B **implies** B^c $\subseteq$ A.

Theorem SUBSET_1:37. $\emptyset$ E $\subseteq$ E.

Theorem SUBSET_1:38. A $\subseteq$ A^c **iff** A = $\emptyset$ E.

Theorem SUBSET_1:39. A^c $\subseteq$ A **iff** A = ΩE.

Theorem SUBSET_1:40. X $\subseteq$ A & X $\subseteq$ A^c **implies** X = $\emptyset$.

Theorem SUBSET_1:41. (A$\cup$B)^c $\subseteq$ A^c & (A$\cup$B)^c $\subseteq$ B^c.

Theorem SUBSET_1:42. A^c $\subseteq$ (A$\cap$B)^c & B^c $\subseteq$ (A$\cap$B)^c.

Theorem SUBSET_1:43. A misses B **iff** A $\subseteq$ B^c.

Theorem SUBSET_1:44. A misses B^c **iff** A $\subseteq$ B.

Theorem SUBSET_1:45. A misses A^c.

Theorem SUBSET_1:46. A misses B & A^c misses B^c **implies** A = B^c.

Theorem SUBSET_1:47. A $\subseteq$ B & C misses B **implies** A $\subseteq$ C^c.

Theorem SUBSET_1:48. (**for** a **being** Element **of** A **holds** a $\in$ B) **implies** A $\subseteq$ B.

Theorem SUBSET_1:49. (**for** x **being** Element **of** E **holds** x $\in$ A) **implies** E = A.

Theorem SUBSET_1:50. E $\neq$ $\emptyset$ **implies for** A, B **holds** A = B^c **iff for** x **being** Element **of** E **holds** x $\in$ A **iff not** x $\in$ B.

Theorem SUBSET_1:51. E $\neq$ $\emptyset$ **implies for** A, B **holds** A = B^c **iff for** x **being** Element **of** E **holds not** x $\in$ A **iff** x $\in$ B.

Theorem SUBSET_1:52. E $\neq$ $\emptyset$ **implies for** A, B **holds** A = B^c **iff for** x **being** Element **of** E **holds not** (x $\in$ A **iff** x $\in$ B).

Theorem SUBSET_1:53. x $\in$ A^c **implies not** x $\in$ A.

reserve x1, x2, x3, x4, x5, x6, x7, x8 **for** Element **of** X.

Theorem SUBSET_1:54. X $\neq \emptyset$ **implies** {x1} **is** Subset **of** X.

Theorem SUBSET_1:55. X $\neq \emptyset$ **implies** {x1, x2} **is** Subset **of** X.

Theorem SUBSET_1:56. X $\neq \emptyset$ **implies** {x1, x2, x3} **is** Subset **of** X.

Theorem SUBSET_1:57. X $\neq \emptyset$ **implies** {x1, x2, x3, x4} **is** Subset **of** X.

Theorem SUBSET_1:58. X $\neq \emptyset$ **implies** {x1, x2, x3, x4, x5} **is** Subset **of** X.

Theorem SUBSET_1:59. X $\neq \emptyset$ **implies** {x1, x2, x3, x4, x5, x6} **is** Subset **of** X.

Theorem SUBSET_1:60. X $\neq \emptyset$ **implies** {x1, x2, x3, x4, x5, x6, x7} **is** Subset **of** X.

Theorem SUBSET_1:61. X $\neq \emptyset$ **implies** {x1, x2, x3, x4, x5, x6, x7, x8} **is** Subset **of** X.

reserve x1, x2, x3, x4, x5, x6, x7, x8 **for** Any.

Theorem SUBSET_1:62. x1 $\in$ X **implies** {x1} **is** Subset **of** X.

Theorem SUBSET_1:63. x1 $\in$ X & x2 $\in$ X **implies** {x1, x2} **is** Subset **of** X.

Theorem SUBSET_1:64. x1 $\in$ X & x2 $\in$ X & x3 $\in$ X **implies** {x1, x2, x3} **is** Subset **of** X.

Theorem SUBSET_1:65. x1 $\in$ X & x2 $\in$ X & x3 $\in$ X & x4 $\in$ X **implies** {x1, x2, x3, x4} **is** Subset **of** X.

Theorem SUBSET_1:66. x1 $\in$ X & x2 $\in$ X & x3 $\in$ X & x4 $\in$ X & x5 $\in$ X **implies** {x1, x2, x3, x4, x5} **is** Subset **of** X.

Theorem SUBSET_1:67. x1 $\in$ X & x2 $\in$ X & x3 $\in$ X & x4 $\in$ X & x5 $\in$ X & x6 $\in$ X **implies** {x1, x2, x3, x4, x5, x6} **is** Subset **of** X.

Theorem SUBSET_1:68. x1 $\in$ X & x2 $\in$ X & x3 $\in$ X & x4 $\in$ X & x5 $\in$ X & x6 $\in$ X & x7 $\in$ X **implies** {x1, x2, x3, x4, x5, x6, x7} **is** Subset **of** X.

Theorem SUBSET_1:69. x1 $\in$ X & x2 $\in$ X & x3 $\in$ X & x4 $\in$ X & x5 $\in$ X & x6 $\in$ X & x7 $\in$ X & x8 $\in$ X **implies** {x1, x2, x3, x4, x5, x6, x7, x8} **is** Subset **of** X.

scheme Subset_Ex{A() $\rightarrow$ set, P[Any]}: **ex** X **being** Subset **of** A() **st for** x **holds** x $\in$ X **iff** x $\in$ A() & P[x].

Chapter 8

FUNCT_1

Functions and Their Basic Properties

by

Czesław Byliński [1]

Warsaw University (Białystok)

Summary. The definitions of the mode Function and the graph of a function are introduced. The graph of a function is defined to be identical with the function. The following concepts are also defined: the domain of a function, the range of a function, the identity function, the composition of functions, the 1-1 function, the inverse function, the restriction of a function, the image and the inverse image. Certain basic facts about functions and the notions defined in the article are proved.

The symbols used in this article are introduced in the following vocabularies: FAM_OP, BOOLE, REAL_1, FUNC_REL, and FUNC. The articles TARSKI and BOOLE provide the terminology and notation for this article.

reserve X, X1, X2, Y, Y1, Y2 **for** set, p, x, x1, x2, y, y1, y2, z, z1, z2 **for** Any.

Definition

mode Function → Any **means ex** F **being** set **st it** = F & (**for** p **st** p ∈ F **ex** x, y **st** [x, y] = p) & (**for** x, y1, y2 **st** [x, y1] ∈ F & [x, y2] ∈ F **holds** y1 = y2).

reserve f, f1, f2, g, g1, g2, h **for** Function.

Definition

let f.

[1]Supported by RPBP.III-24.C1.

func graph f → set **means** f = **it**.

Theorem FUNCT_1:1. graph f = f.

Theorem FUNCT_1:2. **for** F **being** set **st** (**for** p **st** p ∈ F **ex** x, y **st** [x, y] = p) & (**for** x, y1, y2 **st** [x, y1] ∈ F & [x, y2] ∈ F **holds** y1 = y2) **ex** f **being** Function **st** graph f = F.

Theorem FUNCT_1:3. p ∈ graph f **implies** **ex** x, y **st** [x, y] = p.

Theorem FUNCT_1:4. [x, y1] ∈ graph f & [x, y2] ∈ graph f **implies** y1 = y2.

Theorem FUNCT_1:5. graph f = graph g **implies** f = g.

scheme GraphFunc{A() → set, P[Any, Any]}: **ex** f **st** **for** x, y **holds** [x, y] ∈ graph f **iff** x ∈ A() & P[x, y] **provided** A: **for** x, y1, y2 **st** P[x, y1] & P[x, y2] **holds** y1 = y2.

Definition

let f.

func dom f → set **means** **for** x **holds** x ∈ **it** **iff** **ex** y **st** [x, y] ∈ graph f.

Theorem FUNCT_1:6. X = dom f **iff** **for** x **holds** x ∈ X **iff** **ex** y **st** [x, y] ∈ graph f.

Definition

let f, x.

assume x ∈ dom f.

func f.x → Any **means** [x, **it**] ∈ graph f.

Theorem FUNCT_1:7. x ∈ dom f **implies** (y = f.x **iff** [x, y] ∈ graph f).

Theorem FUNCT_1:8. [x, y] ∈ graph f **iff** x ∈ dom f & y = f.x.

Theorem FUNCT_1:9. X = dom f & X = dom g & (**for** x **st** x ∈ X **holds** f.x = g.x) **implies** f = g.

Definition

let f.

func rng f → set **means** **for** y **holds** y ∈ **it** **iff** **ex** x **st** x ∈ dom f & y = f.x.

Theorem FUNCT_1:10. Y = rng f **iff** **for** y **holds** y ∈ Y **iff** **ex** x **st** x ∈ dom f & y = f.x.

Theorem FUNCT_1:11. y ∈ rng f **iff** **ex** x **st** x ∈ dom f & y = f.x.

Theorem FUNCT_1:12. x ∈ dom f **implies** f.x ∈ rng f.

Theorem FUNCT_1:13. dom f = ∅ **iff** rng f = ∅.

Theorem FUNCT_1:14. dom f = {x} **implies** rng f = {f.x}.

scheme FuncEx{A() → set, P[Any, Any]}: **ex** f **st** dom f = A() & **for** x **st** x ∈ A() **holds** P[x, f.x] **provided** A: **for** x, y1, y2 **st** x ∈ A() & P[x, y1] & P[x, y2] **holds** y1 = y2 **and** B: **for** x **st** x ∈ A() **ex** y **st** P[x, y].

scheme Lambda{A() → set, F(Any) → Any}: **ex** f **being** Function **st** dom f = A() & **for** x **st** x ∈ A() **holds** f.x = F(x).

Theorem FUNCT_1:15. X ≠ ∅ **implies for** y **ex** f **st** dom f = X & rng f = {y}.

Theorem FUNCT_1:16. (**for** f, g **st** dom f = X & dom g = X **holds** f = g) **implies** X = ∅.

Theorem FUNCT_1:17. dom f = dom g & rng f = {y} & rng g = {y} **implies** f = g.

Theorem FUNCT_1:18. Y ≠ ∅ **or** X = ∅ **implies ex** f **st** X = dom f & rng f ⊆ Y.

Theorem FUNCT_1:19. (**for** y **st** y ∈ Y **ex** x **st** x ∈ dom f & y = f.x) **implies** Y ⊆ rng f.

Definition

let f, g.

func g·f → Function **means** (**for** x **holds** x ∈ dom **it iff** x ∈ dom f & f.x ∈ dom g) & (**for** x **st** x ∈ dom **it holds it**.x = g.(f.x)).

Theorem FUNCT_1:20. h = g·f **iff** (**for** x **holds** x ∈ dom h **iff** x ∈ dom f & f.x ∈ dom g) & (**for** x **st** x ∈ dom h **holds** h.x = g.(f.x)).

Theorem FUNCT_1:21. x ∈ dom (g·f) **iff** x ∈ dom f & f.x ∈ dom g.

Theorem FUNCT_1:22. x ∈ dom (g·f) **implies** (g·f).x = g.(f.x).

Theorem FUNCT_1:23. x ∈ dom f & f.x ∈ dom g **implies** (g·f).x = g.(f.x).

Theorem FUNCT_1:24. dom (g·f) ⊆ dom f.

Theorem FUNCT_1:25. z ∈ rng (g·f) **implies** z ∈ rng g.

Theorem FUNCT_1:26. rng (g·f) ⊆ rng g.

Theorem FUNCT_1:27. rng f ⊆ dom g **iff** dom (g·f) = dom f.

Theorem FUNCT_1:28. dom g ⊆ rng f **implies** rng (g·f) = rng g.

Theorem FUNCT_1:29. rng f = dom g **implies** dom (g·f) = dom f & rng (g·f) = rng g.

Theorem FUNCT_1:30. h·(g·f) = (h·g)·f.

Theorem FUNCT_1:31. rng f ⊆ dom g & x ∈ dom f **implies** (g·f).x = g.(f.x).

Theorem FUNCT_1:32. rng f = dom g & x ∈ dom f **implies** (g·f).x = g.(f.x).

Theorem FUNCT_1:33. rng f ⊆ Y & (**for** g, h **st** dom g = Y & dom h = Y & g·f = h·f **holds** g = h) **implies** Y = rng f.

Definition

let X.

func Id X → Function **means** dom **it** = X & **for** x **st** x ∈ X **holds it**.x = x.

Theorem FUNCT_1:34. f = Id X **iff** dom f = X & **for** x **st** x ∈ X **holds** f.x = x.

Theorem FUNCT_1:35. x ∈ X **implies** (Id X).x = x.

Theorem FUNCT_1:36. dom Id X = X & rng Id X = X.

Theorem FUNCT_1:37. dom (f·(Id X)) = dom f∩X.

Theorem FUNCT_1:38. x ∈ dom f∩X **implies** f.x = (f·(Id X)).x.

Theorem FUNCT_1:39. dom f ⊆ X **implies** f·(Id X) = f.

Theorem FUNCT_1:40. x ∈ dom ((Id Y)·f) **iff** x ∈ dom f & f.x ∈ Y.

Theorem FUNCT_1:41. rng f ⊆ Y **implies** (Id Y)·f = f.

Theorem FUNCT_1:42. f·(Id dom f) = f & (Id rng f)·f = f.

Theorem FUNCT_1:43. (Id X)·(Id Y) = Id (X∩Y).

Theorem FUNCT_1:44. dom f = X & rng f = X & dom g = X & g·f = f **implies** g = Id X.

Definition

let f.

pred f is 1-1 **means for** x1, x2 **st** x1 ∈ dom f & x2 ∈ dom f & f.x1 = f.x2 **holds** x1 = x2.

Theorem FUNCT_1:45. f is 1-1 **iff for** x1, x2 **st** x1 ∈ dom f & x2 ∈ dom f & f.x1 = f.x2 **holds** x1 = x2.

Theorem FUNCT_1:46. f is 1-1 & g is 1-1 **implies** g·f is 1-1.

Theorem FUNCT_1:47. g·f is 1-1 & rng f ⊆ dom g **implies** f is 1-1.

Theorem FUNCT_1:48. g·f is 1-1 & rng f = dom g **implies** f is 1-1 & g is 1-1.

Theorem FUNCT_1:49. f is 1-1 **iff** (**for** g, h **st** rng g ⊆ dom f & rng h ⊆ dom f & dom g = dom h & f·g = f·h **holds** g = h).

Theorem FUNCT_1:50. dom f = X & dom g = X & rng g ⊆ X & f is 1-1 & f·g = f **implies** g = Id X.

Theorem FUNCT_1:51. rng (g·f) = rng g & g is 1-1 **implies** dom g ⊆ rng f.

Theorem FUNCT_1:52. Id X is 1-1.

Theorem FUNCT_1:53. (**ex** g **st** g·f = Id dom f) **implies** f is 1-1.

Definition

let f.

assume f is 1-1.

func f^{-1} → Function **means** dom **it** = rng f & **for** y, x **holds** y ∈ rng f & x = **it**.y **iff** x ∈ dom f & y = f.x.

Theorem FUNCT_1:54. f is 1-1 **implies** (g = f^{-1} **iff** dom g = rng f & **for** y, x **holds** y ∈ rng f & x = g.y **iff** x ∈ dom f & y = f.x).

Theorem FUNCT_1:55. f is 1-1 **implies** rng f = dom (f^{-1}) & dom f = rng (f^{-1}).

Theorem FUNCT_1:56. f is 1-1 & x ∈ dom f **implies** x = (f^{-1}).(f.x) & x = (f^{-1}·f).x.

Theorem FUNCT_1:57. f is 1-1 & y ∈ rng f **implies** y = f.((f^{-1}).y) & y = (f·f^{-1}).y.

Theorem FUNCT_1:58. f is 1-1 **implies** dom (f^{-1}·f) = dom f & rng (f^{-1}·f) = dom f.

Theorem FUNCT_1:59. f is 1-1 **implies** dom (f·f^{-1}) = rng f & rng (f·f^{-1}) = rng f.

Theorem FUNCT_1:60. f is 1-1 & dom f = rng g & rng f = dom g & (**for** x, y **st** x ∈ dom f & y ∈ dom g **holds** f.x = y **iff** g.y = x) **implies** g = f^{-1}.

Theorem FUNCT_1:61. f is 1-1 **implies** $f^{-1}\cdot f$ = Id dom f & $f\cdot f^{-1}$ = Id rng f.

Theorem FUNCT_1:62. f is 1-1 **implies** f^{-1} is 1-1.

Theorem FUNCT_1:63. f is 1-1 & rng f = dom g & g·f = Id dom f **implies** g = f^{-1}.

Theorem FUNCT_1:64. f is 1-1 & rng g = dom f & f·g = Id rng f **implies** g = f^{-1}.

Theorem FUNCT_1:65. f is 1-1 **implies** $(f^{-1})^{-1}$ = f.

Theorem FUNCT_1:66. f is 1-1 & g is 1-1 **implies** $(g\cdot f)^{-1} = f^{-1}\cdot g^{-1}$.

Theorem FUNCT_1:67. $(\text{Id } X)^{-1}$ = (Id X).

Definition

let f, X.

func f↾X → Function **means** dom **it** = dom f∩X & **for** x **st** x ∈ dom **it** **holds** **it**.x = f.x.

Theorem FUNCT_1:68. g = f↾X **iff** dom g = dom f∩X & **for** x **st** x ∈ dom g **holds** g.x = f.x.

Theorem FUNCT_1:69. dom (f↾X) = dom f∩X.

Theorem FUNCT_1:70. x ∈ dom (f↾X) **implies** (f↾X).x = f.x.

Theorem FUNCT_1:71. x ∈ dom f∩X **implies** (f↾X).x = f.x.

Theorem FUNCT_1:72. x ∈ dom f & x ∈ X **implies** (f↾X).x = f.x.

Theorem FUNCT_1:73. x ∈ dom f & x ∈ X **implies** f.x ∈ rng (f↾X).

Theorem FUNCT_1:74. X ⊆ dom f **implies** dom (f↾X) = X.

Theorem FUNCT_1:75. dom (f↾X) ⊆ X.

Theorem FUNCT_1:76. dom (f↾X) ⊆ dom f & rng (f↾X) ⊆ rng f.

Theorem FUNCT_1:77. f↾X = f·(Id X).

Theorem FUNCT_1:78. dom f ⊆ X **implies** f↾X = f.

Theorem FUNCT_1:79. f↾(dom f) = f.

Theorem FUNCT_1:80. (f↾X)↾Y = f↾(X∩Y).

Theorem FUNCT_1:81. (f↾X)↾X = f↾X.

Theorem FUNCT_1:82. X ⊆ Y **implies** (f↾X)↾Y = f↾X & (f↾Y)↾X = f↾X.

Theorem FUNCT_1:83. (g·f)↾X = g·(f↾X).

Theorem FUNCT_1:84. f is 1-1 **implies** f↾X is 1-1.

Definition

let Y, f.

func Y↾f → Function **means** (**for** x **holds** x ∈ dom **it** **iff** x ∈ dom f & f.x ∈ Y) & (**for** x **st** x ∈ dom **it** **holds** **it**.x = f.x).

Theorem FUNCT_1:85. g = Y↾f **iff** (**for** x **holds** x ∈ dom g **iff** x ∈ dom f & f.x ∈ Y) & (**for** x **st** x ∈ dom g **holds** g.x = f.x).

Theorem FUNCT_1:86. x ∈ dom (Y↾f) **iff** x ∈ dom f & f.x ∈ Y.

Theorem FUNCT_1:87. x ∈ dom (Y↾f) **implies** (Y↾f).x = f.x.

Theorem FUNCT_1:88. rng (Y↾f) ⊆ Y.

Theorem FUNCT_1:89. dom (Y↾f) ⊆ dom f & rng (Y↾f) ⊆ rng f.

Theorem FUNCT_1:90. rng (Y↾f) = rng f∩Y.

Theorem FUNCT_1:91. Y ⊆ rng f **implies** rng (Y↾f) = Y.

Theorem FUNCT_1:92. Y↾f = (Id Y)·f.

Theorem FUNCT_1:93. rng f ⊆ Y **implies** Y↾f = f.

Theorem FUNCT_1:94. (rng f)↾f = f.

Theorem FUNCT_1:95. Y↾(X↾f) = (Y∩X)↾f.

Theorem FUNCT_1:96. Y↾(Y↾f) = Y↾f.

Theorem FUNCT_1:97. X ⊆ Y **implies** Y↾(X↾f) = X↾f & X↾(Y↾f) = X↾f.

Theorem FUNCT_1:98. Y↾(g·f) = (Y↾g)·f.

Theorem FUNCT_1:99. f is 1-1 **implies** Y↾f is 1-1.

Theorem FUNCT_1:100. (Y↾f)↾X = Y↾(f↾X).

Definition

let f, X.

func f.:X → set **means for** y **holds** y ∈ **it iff ex** x **st** x ∈ dom f & x ∈ X & y = f.x.

Theorem FUNCT_1:101. Y = f.:X **iff for** y **holds** y ∈ Y **iff ex** x **st** x ∈ dom f & x ∈ X & y = f.x.

Theorem FUNCT_1:102. y ∈ f.:X **iff ex** x **st** x ∈ dom f & x ∈ X & y = f.x.

Theorem FUNCT_1:103. f.:X ⊆ rng f.

Theorem FUNCT_1:104. f.:(X) = f.:(dom f∩X).

Theorem FUNCT_1:105. f.:(dom f) = rng f.

Theorem FUNCT_1:106. f.:X ⊆ f.:(dom f).

Theorem FUNCT_1:107. rng (f↾X) = f.:X.

Theorem FUNCT_1:108. f.:X = ∅ **iff** dom f∩X = ∅.

Theorem FUNCT_1:109. f.:∅ = ∅.

Theorem FUNCT_1:110. X ≠ ∅ & X ⊆ dom f **implies** f.:X ≠ ∅.

Theorem FUNCT_1:111. X1 ⊆ X2 **implies** f.:X1 ⊆ f.:X2.

Theorem FUNCT_1:112. f.:(X1∪X2) = f.:X1∪f.:X2.

Theorem FUNCT_1:113. f.:(X1∩X2) ⊆ f.:X1∩f.:X2.

Theorem FUNCT_1:114. f.:X1∖f.:X2 ⊆ f.:(X1∖X2).

Theorem FUNCT_1:115. (g·f).:X = g.:(f.:X).

Theorem FUNCT_1:116. rng (g·f) = g.(rng f).

Theorem FUNCT_1:117. x ∈ dom f **implies** f.{x} = {f.x}.

Theorem FUNCT_1:118. x1 ∈ dom f & x2 ∈ dom f **implies** f.{x1, x2} = {f.x1, f.x2}.

Theorem FUNCT_1:119. (f↾Y).X ⊆ f.X.

Theorem FUNCT_1:120. (Y↾f).X ⊆ f.X.

Theorem FUNCT_1:121. f is 1-1 **implies** f.(X1∩X2) = f.X1∩f.X2.

Theorem FUNCT_1:122. (**for** X1, X2 **holds** f.(X1∩X2) = f.X1∩f.X2) **implies** f is 1-1.

Theorem FUNCT_1:123. f is 1-1 **implies** f.(X1∖X2) = f.X1∖f.X2.

Theorem FUNCT_1:124. (**for** X1, X2 **holds** f.(X1∖X2) = f.X1∖f.X2) **implies** f is 1-1.

Theorem FUNCT_1:125. X∩Y = ∅ & f is 1-1 **implies** f.X∩f.Y = ∅.

Theorem FUNCT_1:126. (Y↾f).X = Y∩f.X.

Definition

let f, Y.

func f^{-1}Y → set **means for** x **holds** x ∈ **it iff** x ∈ dom f & f.x ∈ Y.

Theorem FUNCT_1:127. X = f^{-1}Y **iff for** x **holds** x ∈ X **iff** x ∈ dom f & f.x ∈ Y.

Theorem FUNCT_1:128. x ∈ f^{-1}Y **iff** x ∈ dom f & f.x ∈ Y.

Theorem FUNCT_1:129. f^{-1}Y ⊆ dom f.

Theorem FUNCT_1:130. f^{-1}Y = f^{-1}(rng f∩Y).

Theorem FUNCT_1:131. f^{-1}(rng f) = dom f.

Theorem FUNCT_1:132. f^{-1}∅ = ∅.

Theorem FUNCT_1:133. f^{-1}Y = ∅ **iff** rng f∩Y = ∅.

Theorem FUNCT_1:134. Y ⊆ rng f **implies** (f^{-1}Y = ∅ **iff** Y = ∅).

Theorem FUNCT_1:135. Y1 ⊆ Y2 **implies** f^{-1}Y1 ⊆ f^{-1}Y2.

Theorem FUNCT_1:136. f^{-1}(Y1∪Y2) = f^{-1}Y1∪f^{-1}Y2.

Theorem FUNCT_1:137. f^{-1}(Y1∩Y2) = f^{-1}Y1∩f^{-1}Y2.

Theorem FUNCT_1:138. f^{-1}(Y1∖Y2) = f^{-1}Y1∖f^{-1}Y2.

Theorem FUNCT_1:139. $(f↾X)^{-1}$Y = X∩(f^{-1}Y).

Theorem FUNCT_1:140. $(g·f)^{-1}$Y = f^{-1}(g^{-1}Y).

Theorem FUNCT_1:141. dom (g·f) = f^{-1}(dom g).

Theorem FUNCT_1:142. y ∈ rng f **iff** f^{-1}{y} ≠ ∅.

Theorem FUNCT_1:143. (**for** y **st** y ∈ Y **holds** f^{-1}{y} ≠ ∅) **implies** Y ⊆ rng f.

Theorem FUNCT_1:144. (**for** y **st** y ∈ rng f **ex** x **st** f^{-1}{y} = {x}) **iff** f is 1-1.

Theorem FUNCT_1:145. f.(f^{-1}Y) ⊆ Y.

Theorem FUNCT_1:146. X ⊆ dom f **implies** X ⊆ f^{-1}(f.X).

Theorem FUNCT_1:147. Y ⊆ rng f **implies** f.(f^{-1}Y) = Y.

Theorem FUNCT_1:148. $f.(f^{-1}Y) = Y \cap f.(\mathsf{dom}\ f)$.

Theorem FUNCT_1:149. $f.(X \cap f^{-1}Y) \subseteq (f.X) \cap Y$.

Theorem FUNCT_1:150. $f.(X \cap f^{-1}Y) = (f.X) \cap Y$.

Theorem FUNCT_1:151. $X \cap f^{-1}Y \subseteq f^{-1}(f.X \cap Y)$.

Theorem FUNCT_1:152. f is 1-1 **implies** $f^{-1}(f.X) \subseteq X$.

Theorem FUNCT_1:153. (**for** X **holds** $f^{-1}(f.X) \subseteq X$) **implies** f is 1-1.

Theorem FUNCT_1:154. f is 1-1 **implies** $f.X = (f^{-1})^{-1}X$.

Theorem FUNCT_1:155. f is 1-1 **implies** $f^{-1}Y = (f^{-1}).Y$.

Theorem FUNCT_1:156. Y = rng f & dom g = Y & dom h = Y & $g \cdot f = h \cdot f$ **implies** g = h.

Theorem FUNCT_1:157. $f.X1 \subseteq f.X2$ & $X1 \subseteq$ dom f & f is 1-1 **implies** $X1 \subseteq X2$.

Theorem FUNCT_1:158. $f^{-1}Y1 \subseteq f^{-1}Y2$ & $Y1 \subseteq$ rng f **implies** $Y1 \subseteq Y2$.

Theorem FUNCT_1:159. f is 1-1 **iff for** y **ex** x **st** $f^{-1}\{y\} \subseteq \{x\}$.

Theorem FUNCT_1:160. rng f $\subseteq$ dom g **implies** $f^{-1}X \subseteq (g \cdot f)^{-1}(g.X)$.

Chapter 9

FUNCT_2

Functions from a Set to a Set.

by

Czesław Byliński [1]

Warsaw University (Białystok)

Summary. The article is a continuation of *Functions and Their Basic Properties* (FUNCT_1). We define the following concepts: a function from a set X into a set Y, denoted by "Function of X,Y", the set of all functions from a set X into a set Y, denoted by Funcs(X,Y), and the permutation of a set (mode Permutation of X, where X is a set). Theorems and schemes included in the article are reformulations of the theorems of FUNCT_1 in the new terminology. Also some basic facts about functions of two variables are proved.

The symbols used in this article are introduced in the following vocabularies: BOOLE, FUNC_REL, REAL_1, FUNC, and FUNC2. The terminology and notation used in this article have been introduced in the following articles: TARSKI, BOOLE, and FUNCT_1.

reserve P, Q, X, X1, X2, Y, Y1, Y2, Z **for** set.

reserve p, q, x, x1, x2, y, y1, y2, z, z1, z2 **for** Any.

Definition

let X, Y.

assume $Y = \emptyset$ **implies** $X = \emptyset$.

mode Function **of** X, Y $\rightarrow$ Function **means** X = dom **it** & rng **it** $\subseteq$ Y.

[1]Supported by RPBP.III-24.C1.

Theorem FUNCT_2:1. (Y = ∅ **implies** X = ∅) **implies** **for** f **being** Function **holds** f **is** Function **of** X, Y **iff** X = dom f & rng f ⊆ Y.

Theorem FUNCT_2:2. **for** f **being** Function **of** X, Y **st** Y = ∅ **implies** X = ∅ **holds** X = dom f & rng f ⊆ Y.

Theorem FUNCT_2:3. **for** f **being** Function **holds** f **is** Function **of** dom f, rng f.

Theorem FUNCT_2:4. **for** f **being** Function **st** rng f ⊆ Y **holds** f **is** Function **of** dom f, Y.

Theorem FUNCT_2:5. **for** f **being** Function **st** dom f = X & **for** x **st** x ∈ X **holds** f.x ∈ Y **holds** f **is** Function **of** X, Y.

Theorem FUNCT_2:6. **for** f **being** Function **of** X, Y **st** Y ≠ ∅ & x ∈ X **holds** f.x ∈ rng f.

Theorem FUNCT_2:7. **for** f **being** Function **of** X, Y **st** Y ≠ ∅ & x ∈ X **holds** f.x ∈ Y.

Theorem FUNCT_2:8. **for** f **being** Function **of** X, Y **st** (Y = ∅ **implies** X = ∅) & rng f ⊆ Z **holds** f **is** Function **of** X, Z.

Theorem FUNCT_2:9. **for** f **being** Function **of** X, Y **st** (Y = ∅ **implies** X = ∅) & Y ⊆ Z **holds** f **is** Function **of** X, Z.

scheme FuncEx1{X() → set, Y() → set, P[Any, Any]}: **ex** f **being** Function **of** X(), Y() **st** **for** x **st** x ∈ X() **holds** P[x, f.x] **provided** A1: **for** x **st** x ∈ X() **ex** y **st** y ∈ Y() & P[x, y] **and** A2: **for** x, y1, y2 **st** x ∈ X() & P[x, y1] & P[x, y2] **holds** y1 = y2.

scheme Lambda1{X() → set, Y() → set, F(Any) → Any}: **ex** f **being** Function **of** X(), Y() **st** **for** x **st** x ∈ X() **holds** f.x = F(x) **provided** A: **for** x **st** x ∈ X() **holds** F(x) ∈ Y().

Definition

let X, Y.

func Funcs (X, Y) → set **means** x ∈ **it** **iff** **ex** f **being** Function **st** x = f & dom f = X & rng f ⊆ Y.

Theorem FUNCT_2:10. **for** F **being** set **holds** F = Funcs (X, Y) **iff** **for** x **holds** x ∈ F **iff** **ex** f **being** Function **st** x = f & dom f = X & rng f ⊆ Y.

Theorem FUNCT_2:11. **for** f **being** Function **of** X, Y **st** Y = ∅ **implies** X = ∅ **holds** f ∈ Funcs (X, Y).

Theorem FUNCT_2:12. **for** f **being** Function **of** X, X **holds** f ∈ Funcs (X, X).

Theorem FUNCT_2:13. **for** f **being** Function **of** ∅, X **holds** f ∈ Funcs (∅, X).

Theorem FUNCT_2:14. X ≠ ∅ **implies** Funcs (X, ∅) = ∅.

Theorem FUNCT_2:15. Funcs (X, Y) = ∅ **implies** X ≠ ∅ & Y = ∅.

Theorem FUNCT_2:16. **for** f **being** Function **of** X, Y **st** Y ≠ ∅ & **for** y **st** y ∈ Y **ex** x **st** x ∈ X & y = f.x **holds** rng f = Y.

Theorem FUNCT_2:17. **for** f **being** Function **of** X, Y **st** y ∈ Y & rng f = Y **ex** x **st** x ∈ X & f.x = y.

Theorem FUNCT_2:18. **for** f1, f2 **being** Function **of** X, Y **st** Y ≠ ∅ & **for** x **st** x ∈ X **holds** f1.x = f2.x **holds** f1 = f2.

Theorem FUNCT_2:19. **for** f **being** Function **of** X, Y **for** g **being** Function **of** Y, Z **st** (Z = ∅ **implies** Y = ∅) & (Y = ∅ **implies** X = ∅) **holds** g·f **is** Function **of** X, Z.

Theorem FUNCT_2:20. **for** f **being** Function **of** X, Y **for** g **being** Function **of** Y, Z **st** Y ≠ ∅ & Z ≠ ∅ & rng f = Y & rng g = Z **holds** rng (g·f) = Z.

Theorem FUNCT_2:21. **for** f **being** Function **of** X, Y **for** g **being** Function **of** Y, Z **st** Y ≠ ∅ & Z ≠ ∅ & x ∈ X **holds** (g·f).x = g.(f.x).

Theorem FUNCT_2:22. **for** f **being** Function **of** X, Y **st** Y ≠ ∅ **holds** rng f = Y **iff** **for** Z **st** Z ≠ ∅ **for** g, h **being** Function **of** Y, Z **st** g·f = h·f **holds** g = h.

Theorem FUNCT_2:23. **for** f **being** Function **of** X, Y **st** Y = ∅ **implies** X = ∅ **holds** f·(Id X) = f & (Id Y)·f = f.

Theorem FUNCT_2:24. **for** f **being** Function **of** X, Y **for** g **being** Function **of** Y, X **st** Y ≠ ∅ & f·g = Id Y **holds** rng f = Y.

Theorem FUNCT_2:25. **for** f **being** Function **of** X, Y **st** Y = ∅ **implies** X = ∅ **holds** f is 1-1 **iff** **for** x1, x2 **st** x1 ∈ X & x2 ∈ X & f.x1 = f.x2 **holds** x1 = x2.

Theorem FUNCT_2:26. **for** f **being** Function **of** X, Y **for** g **being** Function **of** Y, Z **st** (Z = ∅ **implies** Y = ∅) & (Y = ∅ **implies** X = ∅) & g·f is 1-1 **holds** f is 1-1.

Theorem FUNCT_2:27. **for** f **being** Function **of** X, Y **st** X ≠ ∅ & Y ≠ ∅ **holds** f is 1-1 **iff** **for** Z **for** g, h **being** Function **of** Z, X **st** f·g = f·h **holds** g = h.

Theorem FUNCT_2:28. **for** f **being** Function **of** X, Y **for** g **being** Function **of** Y, Z **st** Z ≠ ∅ & Y ≠ ∅ & rng (g·f) = Z & g is 1-1 **holds** rng f = Y.

Theorem FUNCT_2:29. **for** f **being** Function **of** X, Y **for** g **being** Function **of** Y, X **st** X ≠ ∅ & Y ≠ ∅ & g·f = Id X **holds** f is 1-1 & rng g = X.

Theorem FUNCT_2:30. **for** f **being** Function **of** X, Y **for** g **being** Function **of** Y, Z **st** (Z = ∅ **implies** Y = ∅) & g·f is 1-1 & rng f = Y **holds** f is 1-1 & g is 1-1.

Theorem FUNCT_2:31. **for** f **being** Function **of** X, Y **st** f is 1-1 & (X = ∅ **iff** Y = ∅) & rng f = Y **holds** f^{-1} **is** Function **of** Y, X.

Theorem FUNCT_2:32. **for** f **being** Function **of** X, Y **st** Y ≠ ∅ & f is 1-1 & x ∈ X **holds** (f^{-1}).(f.x) = x.

Theorem FUNCT_2:33. **for** f **being** Function **of** X, Y **st** rng f = Y & f is 1-1 & y ∈ Y **holds** f.((f^{-1}).y) = y.

Theorem FUNCT_2:34. **for** f **being** Function **of** X, Y **for** g **being** Function **of** Y, X **st** X ≠ ∅ & Y ≠ ∅ & rng f = Y & f is 1-1 & **for** y, x **holds** y ∈ Y & g.y = x **iff** x ∈ X & f.x = y **holds** g = f^{-1}.

Theorem FUNCT_2:35. **for** f **being** Function **of** X, Y **st** Y ≠ ∅ & rng f = Y & f is 1-1 **holds** f^{-1}·f = Id X & f·f^{-1} = Id Y.

Theorem FUNCT_2:36. **for** f **being** Function **of** X, Y **for** g **being** Function **of** Y, X

st X ≠ ∅ & Y ≠ ∅ & rng f = Y & g·f = Id X & f is 1-1 **holds** g = f^{-1}.

Theorem FUNCT_2:37. **for** f **being** Function **of** X, Y **st** Y ≠ ∅ & **ex** g **being** Function **of** Y, X **st** g·f = Id X **holds** f is 1-1.

Theorem FUNCT_2:38. **for** f **being** Function **of** X, Y **st** (Y = ∅ **implies** X = ∅) & Z ⊆ X **holds** f↾Z **is** Function **of** Z, Y.

Theorem FUNCT_2:39. **for** f **being** Function **of** X, Y **st** Y ≠ ∅ & x ∈ X & x ∈ Z **holds** (f↾Z).x = f.x.

Theorem FUNCT_2:40. **for** f **being** Function **of** X, Y **st** (Y = ∅ **implies** X = ∅) & X ⊆ Z **holds** f↾Z = f.

Theorem FUNCT_2:41. **for** f **being** Function **of** X, Y **st** Y ≠ ∅ & x ∈ X & f.x ∈ Z **holds** (Z↾f).x = f.x.

Theorem FUNCT_2:42. **for** f **being** Function **of** X, Y **st** (Y = ∅ **implies** X = ∅) & Y ⊆ Z **holds** Z↾f = f.

Theorem FUNCT_2:43. **for** f **being** Function **of** X, Y **st** Y ≠ ∅ **for** y **holds** y ∈ f**.**P **iff ex** x **st** x ∈ X & x ∈ P & y = f.x.

Theorem FUNCT_2:44. **for** f **being** Function **of** X, Y **st** Y = ∅ **implies** X = ∅ **holds** f**.**P ⊆ Y.

Theorem FUNCT_2:45. **for** f **being** Function **of** X, Y **st** Y = ∅ **implies** X = ∅ **holds** f**.**X = rng f.

Theorem FUNCT_2:46. **for** f **being** Function **of** X, Y **st** Y ≠ ∅ **for** x **holds** x ∈ f^{-1}Q **iff** x ∈ X & f.x ∈ Q.

Theorem FUNCT_2:47. **for** f **being** Function **of** X, Y **st** Y = ∅ **implies** X = ∅ **holds** f^{-1}Q ⊆ X.

Theorem FUNCT_2:48. **for** f **being** Function **of** X, Y **st** Y = ∅ **implies** X = ∅ **holds** f^{-1}Y = X.

Theorem FUNCT_2:49. **for** f **being** Function **of** X, Y **st** Y ≠ ∅ **holds** (**for** y **st** y ∈ Y **holds** f^{-1}{y} ≠ ∅) **iff** rng f = Y.

Theorem FUNCT_2:50. **for** f **being** Function **of** X, Y **st** (Y = ∅ **implies** X = ∅) & P ⊆ X **holds** P ⊆ f^{-1}(f**.**P).

Theorem FUNCT_2:51. **for** f **being** Function **of** X, Y **st** Y = ∅ **implies** X = ∅ **holds** f^{-1}(f**.**X) = X.

Theorem FUNCT_2:52. **for** f **being** Function **of** X, Y **st** (Y = ∅ **implies** X = ∅) & rng f = Y **holds** f**.**(f^{-1}Y) = Y.

Theorem FUNCT_2:53. **for** f **being** Function **of** X, Y **for** g **being** Function **of** Y, Z **st** (Z = ∅ **implies** Y = ∅) & (Y = ∅ **implies** X = ∅) **holds** f^{-1}Q ⊆ $(g·f)^{-1}$(g**.**Q).

Theorem FUNCT_2:54. **for** f **being** Function **of** ∅, Y **holds** dom f = ∅ & rng f = ∅.

Theorem FUNCT_2:55. **for** f **being** Function **st** dom f = ∅ **holds** f **is** Function **of** ∅, Y.

Theorem FUNCT_2:56. **for** f1 **being** Function **of** ∅, Y1 **for** f2 **being** Function **of** ∅, Y2 **holds** f1 = f2.

Theorem FUNCT_2:57. **for** f **being** Function **of** ∅, Y **for** g **being** Function **of** Y, Z **st** Z = ∅ **implies** Y = ∅ **holds** g·f **is** Function **of** ∅, Z.

Theorem FUNCT_2:58. **for** f **being** Function **of** ∅, Y **holds** f is 1-1.

Theorem FUNCT_2:59. **for** f **being** Function **of** ∅, Y **holds** f.P = ∅.

Theorem FUNCT_2:60. **for** f **being** Function **of** ∅, Y **holds** $f^{-1}Q = \emptyset$.

Theorem FUNCT_2:61. **for** f **being** Function **of** {x}, Y **st** Y ≠ ∅ **holds** f.x ∈ Y.

Theorem FUNCT_2:62. **for** f **being** Function **of** {x}, Y **st** Y ≠ ∅ **holds** rng f = {f.x}.

Theorem FUNCT_2:63. **for** f **being** Function **of** {x}, Y **st** Y ≠ ∅ **holds** f is 1-1.

Theorem FUNCT_2:64. **for** f **being** Function **of** {x}, Y **st** Y ≠ ∅ **holds** f.P ⊆ {f.x}.

Theorem FUNCT_2:65. **for** f **being** Function **of** X, {y} **st** x ∈ X **holds** f.x = y.

Theorem FUNCT_2:66. **for** f1, f2 **being** Function **of** X, {y} **holds** f1 = f2.

Definition

let X.

let f, g **being** Function **of** X, X.

redefine

func g·f → Function **of** X, X.

Definition

let X.

redefine

func Id X → Function **of** X, X.

Theorem FUNCT_2:67. **for** f **being** Function **of** X, X **holds** dom f = X & rng f ⊆ X.

Theorem FUNCT_2:68. **for** f **being** Function **st** dom f = X & rng f ⊆ X **holds** f **is** Function **of** X, X.

Theorem FUNCT_2:69. **for** f **being** Function **of** X, X **st** x ∈ X **holds** f.x ∈ X.

Theorem FUNCT_2:70. **for** f, g **being** Function **of** X, X **st** x ∈ X **holds** (g·f).x = g.(f.x).

Theorem FUNCT_2:71. **for** f **being** Function **of** X, X **for** g **being** Function **of** X, Y **st** Y ≠ ∅ & x ∈ X **holds** (g·f).x = g.(f.x).

Theorem FUNCT_2:72. **for** f **being** Function **of** X, Y **for** g **being** Function **of** Y, Y **st** Y ≠ ∅ & x ∈ X **holds** (g·f).x = g.(f.x).

Theorem FUNCT_2:73. **for** f, g **being** Function **of** X, X **st** rng f = X & rng g = X **holds** rng (g·f) = X.

Theorem FUNCT_2:74. **for** f **being** Function **of** X, X **holds** f·(Id X) = f & (Id X)·f = f.

Theorem FUNCT_2:75. **for** f, g **being** Function **of** X, X **st** g·f = f & rng f = X **holds** g = Id X.

Theorem FUNCT_2:76. **for** f, g **being** Function **of** X, X **st** f·g = f & f is 1-1 **holds** g = Id X.

Theorem FUNCT_2:77. **for** f **being** Function **of** X, X **holds** f is 1-1 **iff for** x1, x2 **st** x1 ∈ X & x2 ∈ X & f.x1 = f.x2 **holds** x1 = x2.

Theorem FUNCT_2:78. **for** f **being** Function **of** X, X **holds** f**.**P ⊆ X.

Definition

let X.

let f **be** Function **of** X, X.

let P.

redefine

func f**.**P → Subset **of** X.

Theorem FUNCT_2:79. **for** f **being** Function **of** X, X **holds** f**.**X = rng f.

Theorem FUNCT_2:80. **for** f **being** Function **of** X, X **holds** f^{-1}Q ⊆ X.

Definition

let X.

let f **be** Function **of** X, X.

let Q.

redefine

func f^{-1}Q → Subset **of** X.

Theorem FUNCT_2:81. **for** f **being** Function **of** X, X **st** rng f = X **holds** f**.**(f^{-1}X) = X.

Theorem FUNCT_2:82. **for** f **being** Function **of** X, X **holds** f^{-1}(f**.**X) = X.

Definition

let X.

mode Permutation **of** X → Function **of** X, X **means it** is 1-1 & rng **it** = X.

Theorem FUNCT_2:83. **for** f **being** Function **of** X, X **holds** f **is** Permutation **of** X **iff** f is 1-1 & rng f = X.

Theorem FUNCT_2:84. **for** f **being** Permutation **of** X **holds** f is 1-1 & rng f = X.

Theorem FUNCT_2:85. **for** f **being** Permutation **of** X **for** x1, x2 **st** x1 ∈ X & x2 ∈ X & f.x1 = f.x2 **holds** x1 = x2.

Definition

let X.

let f, g **be** Permutation **of** X.

redefine

func g·f → Permutation **of** X.

Definition

let X.

redefine

func Id X → Permutation **of** X.

Definition

let X.

let f **be** Permutation **of** X.

redefine

func f^{-1} → Permutation **of** X.

Theorem FUNCT_2:86. **for** f, g **being** Permutation **of** X **st** g·f = g **holds** f = Id X.

Theorem FUNCT_2:87. **for** f, g **being** Permutation **of** X **st** g·f = Id X **holds** g = f^{-1}.

Theorem FUNCT_2:88. **for** f **being** Permutation **of** X **holds** (f^{-1})·f = Id X & f·(f^{-1}) = Id X.

Theorem FUNCT_2:89. **for** f **being** Permutation **of** X **holds** $(f^{-1})^{-1}$ = f.

Theorem FUNCT_2:90. **for** f, g **being** Permutation **of** X **holds** $(g \cdot f)^{-1} = f^{-1} \cdot g^{-1}$.

Theorem FUNCT_2:91. **for** f **being** Permutation **of** X **st** P∩Q = ∅ **holds** f.P∩f.Q = ∅.

Theorem FUNCT_2:92. **for** f **being** Permutation **of** X **st** P ⊆ X **holds** f.(f^{-1}P) = P & f^{-1}(f.P) = P.

Theorem FUNCT_2:93. **for** f **being** Permutation **of** X **holds** f.P = $(f^{-1})^{-1}$P & f^{-1}P = (f^{-1}).P.

reserve C, D, E **for** DOMAIN.

Definition

let X, D, E.

let f **be** Function **of** X, D.

let g **be** Function **of** D, E.

redefine

func g·f → Function **of** X, E.

Definition

let X, D.

redefine

mode Function **of** X, D **means** X = dom **it** & rng **it** ⊆ D.

Theorem FUNCT_2:94. **for** f **being** Function **of** X, D **holds** dom f = X & rng f ⊆ D.

Theorem FUNCT_2:95. **for** f **being** Function **st** dom f = X & rng f ⊆ D **holds** f **is** Function **of** X, D.

Theorem FUNCT_2:96. **for** f **being** Function **of** X, D **st** x ∈ X **holds** f.x ∈ D.

Theorem FUNCT_2:97. **for** f **being** Function **of** {x}, D **holds** f.x ∈ D.

Theorem FUNCT_2:98. **for** f1, f2 **being** Function **of** X, D **st** **for** x **st** x ∈ X **holds** f1.x = f2.x **holds** f1 = f2.

Theorem FUNCT_2:99. **for** f **being** Function **of** X, D **for** g **being** Function **of** D, E **st** x ∈ X **holds** (g·f).x = g.(f.x).

Theorem FUNCT_2:100. **for** f **being** Function **of** X, D **holds** f·(Id X) = f & (Id D)·f = f.

Theorem FUNCT_2:101. **for** f **being** Function **of** X, D **holds** f is 1-1 **iff** **for** x1, x2 **st** x1 ∈ X & x2 ∈ X & f.x1 = f.x2 **holds** x1 = x2.

Theorem FUNCT_2:102. **for** f **being** Function **of** X, D **for** y **holds** y ∈ f.P **iff** **ex** x **st** x ∈ X & x ∈ P & y = f.x.

Theorem FUNCT_2:103. **for** f **being** Function **of** X, D **holds** f.P ⊆ D.

Definition

let X, D.

let f **be** Function **of** X, D.

let P.

redefine

func f.P → Subset **of** D.

Theorem FUNCT_2:104. **for** f **being** Function **of** X, D **holds** f.X = rng f.

Theorem FUNCT_2:105. **for** f **being** Function **of** X, D **st** f.X = D **holds** rng (f) = D.

Theorem FUNCT_2:106. **for** f **being** Function **of** X, D **for** x **holds** x ∈ f^{-1}Q **iff** x ∈ X & f.x ∈ Q.

Theorem FUNCT_2:107. **for** f **being** Function **of** X, D **holds** f^{-1}Q ⊆ X.

Definition

let X, D.

let f **be** Function **of** X, D.

let Q.

redefine

func f^{-1}Q → Subset **of** X.

Theorem FUNCT_2:108. **for** f **being** Function **of** X, D **holds** f^{-1}D = X.

Theorem FUNCT_2:109. **for** f **being** Function **of** X, D **holds** (**for** y **st** y ∈ D **holds** f^{-1}{y} ≠ ∅) **iff** rng f = D.

Theorem FUNCT_2:110. **for** f **being** Function **of** X, D **holds** f^{-1}(f.X) = X.

Theorem FUNCT_2:111. **for** f **being** Function **of** X, D **st** rng f = D **holds** f.(f^{-1}D) = D.

Theorem FUNCT_2:112. **for** f **being** Function **of** X, D **for** g **being** Function **of** D, E **holds** f^{-1}Q ⊆ $(g·f)^{-1}$(g.Q).

reserve c, c1, c2 **for** Element **of** C.

reserve d, d1, d2 **for** Element **of** D.

Definition

let C, D.

let f **be** Function **of** C, D.

let c.

redefine

func f.c → Element **of** D.

scheme FuncExD{C() → DOMAIN, D() → DOMAIN, P[Any, Any]}: **ex** f **being** Function **of** C(), D() **st for** x **being** Element **of** C() **holds** P[x, f.x] **provided** A1: **for** x **being** Element **of** C() **ex** y **being** Element **of** D() **st** P[x, y] **and** A2: **for** x **being** (Element **of** C()), y1, y2 **being** Element **of** D() **st** P[x, y1] & P[x, y2] **holds** y1 = y2.

scheme LambdaD{C() → DOMAIN, D() → DOMAIN, F((Element **of** C())) → Element **of** D()}: **ex** f **being** Function **of** C(), D() **st for** x **being** Element **of** C() **holds** f.x = F(x).

Theorem FUNCT_2:113. **for** f1, f2 **being** Function **of** C, D **st for** c **holds** f1.c = f2.c **holds** f1 = f2.

Theorem FUNCT_2:114. (Id C).c = c.

Theorem FUNCT_2:115. **for** f **being** Function **of** C, D **for** g **being** Function **of** D, E **holds** (g·f).c = g.(f.c).

Theorem FUNCT_2:116. **for** f **being** Function **of** C, D **for** d **holds** d ∈ f.P **iff ex** c **st** c ∈ P & d = f.c.

Theorem FUNCT_2:117. **for** f **being** Function **of** C, D **for** c **holds** c ∈ f^{-1}Q **iff** f.c ∈ Q.

Theorem FUNCT_2:118. **for** f1, f2 **being** Function **of** ⟦X, Y⟧, Z **st** Z ≠ ∅ & **for** x, y **st** x ∈ X & y ∈ Y **holds** f1.[x, y] = f2.[x, y] **holds** f1 = f2.

Theorem FUNCT_2:119. **for** f **being** Function **of** ⟦X, Y⟧, Z **st** x ∈ X & y ∈ Y & Z ≠ ∅ **holds** f.[x, y] ∈ Z.

scheme FuncEx2{X() → set, Y() → set, Z() → set, P[Any, Any, Any]}: **ex** f **being** Function **of** ⟦X(), Y()⟧, Z() **st for** x, y **st** x ∈ X() & y ∈ Y() **holds** P[x, y, f.[x, y]] **provided** A1: **for** x, y **st** x ∈ X() & y ∈ Y() **ex** z **st** z ∈ Z() & P[x, y, z] **and** A2: **for** x, y, z1, z2 **st** x ∈ X() & y ∈ Y() & P[x, y, z1] & P[x, y, z2] **holds** z1 = z2.

scheme Lambda2{X() → set, Y() → set, Z() → set, F(Any, Any) → Any}: **ex** f **being** Function **of** ⟦X(), Y()⟧, Z() **st for** x, y **st** x ∈ X() & y ∈ Y() **holds** f.[x, y] = F(x, y) **provided** A: **for** x, y **st** x ∈ X() & y ∈ Y() **holds** F(x, y) ∈ Z().

Theorem FUNCT_2:120. **for** f1, f2 **being** Function **of** ⟦C, D⟧, E **st for** c, d **holds** f1.[c, d] = f2.[c, d] **holds** f1 = f2.

scheme FuncEx2D{X() → DOMAIN, Y() → DOMAIN, Z() → DOMAIN, P[Any, Any, Any]}: **ex** f **being** Function **of** ⟦X(), Y()⟧, Z() **st for** x **being** Element **of** X() **for** y **being**

Element **of** Y() **holds** P[x, y, f.[x, y]] **provided** A1: **for** x **being** Element **of** X() **for** y **being** Element **of** Y() **ex** z **being** Element **of** Z() **st** P[x, y, z] **and** A2: **for** x **being** Element **of** X() **for** y **being** Element **of** Y() **for** z1, z2 **being** Element **of** Z() **st** P[x, y, z1] & P[x, y, z2] **holds** z1 = z2.

scheme Lambda2D{X() → DOMAIN, Y() → DOMAIN, Z() → DOMAIN, F((Element **of** X()), Element **of** Y()) → Element **of** Z()}: **ex** f **being** Function **of** ⟦X(), Y()⟧, Z() **st** **for** x **being** Element **of** X() **for** y **being** Element **of** Y() **holds** f.[x, y] = F(x, y).

Chapter 10

FUNCT_3

Basic Functions and Operations on Functions

by

Czesław Byliński [1]

Warsaw University (Białystok)

Summary. We define the following mappings: the characteristic function of a subset of a set, the inclusion function (injection or embedding), the projections from a cartesian product onto its arguments and diagonal function (inclusion of a set into its cartesian square). Some operations on functions are also defined: the products of two functions (the complex function and the more general product-function), the function induced on power sets by the image and inverse-image. Some simple propositions related to the introduced notions are proved.

The symbols used in this article are introduced in the following vocabularies: BOOLE, BINOP, FUNC, FUNC_REL, REAL_1, FUNC3, and FAM_OP. The terminology and notation used in this article have been introduced in the following articles: TARSKI, BOOLE, FUNCT_1, and FUNCT_2.

reserve p, q, x, x1, x2, y, y1, y2, z, z1, z2 **for** Any.

reserve A, B, V, X, X1, X2, Y, Y1, Y2, Z, P **for** set.

reserve C, C1, C2, D, D1, D2 **for** DOMAIN.

Theorem FUNCT_3:1. A ⊆ Y **implies** Id A = (Id Y)↾A.

Theorem FUNCT_3:2. **for** f, g **being** Function **st** X ⊆ dom (g·f) **holds** f.X ⊆ dom g.

[1]Supported by RPBP.III-24.C1.

Theorem FUNCT_3:3. **for** f, g **being** Function **st** X ⊆ dom f & f.X ⊆ dom g **holds** X ⊆ dom (g·f).

Theorem FUNCT_3:4. **for** f, g **being** Function **st** Y ⊆ rng (g·f) & g is 1-1 **holds** g^{-1}Y ⊆ rng f.

Theorem FUNCT_3:5. **for** f, g **being** Function **st** Y ⊆ rng g & g^{-1}Y ⊆ rng f **holds** Y ⊆ rng (g·f).

scheme FuncEx_3{A() → set, B() → set, P[Any, Any, Any]}: **ex** f **being** Function **st** dom f = ⟦A(), B()⟧ & **for** x, y **st** x ∈ A() & y ∈ B() **holds** P[x, y, f.[x, y]] **provided** A: **for** x, y, z1, z2 **st** x ∈ A() & y ∈ B() & P[x, y, z1] & P[x, y, z2] **holds** z1 = z2 **and** B: **for** x, y **st** x ∈ A() & y ∈ B() **ex** z **st** P[x, y, z].

scheme Lambda_3{A() → set, B() → set, F(Any, Any) → Any}: **ex** f **being** Function **st** dom f = ⟦A(), B()⟧ & **for** x, y **st** x ∈ A() & y ∈ B() **holds** f.[x, y] = F(x, y).

Theorem FUNCT_3:6. **for** f, g **being** Function **st** dom f = ⟦X, Y⟧ & dom g = ⟦X, Y⟧ & **for** x, y **st** x ∈ X & y ∈ Y **holds** f.[x, y] = g.[x, y] **holds** f = g.

Definition

let f **be** Function.

func.f → Function **means** dom **it** = bool dom f & **for** X **st** X ∈ bool dom f **holds** **it**.X = f.X.

Theorem FUNCT_3:7. **for** f, g **being** Function **holds** g =.f **iff** dom g = bool dom f & **for** X **st** X ∈ bool dom f **holds** g.X = f.X.

Theorem FUNCT_3:8. **for** f **being** Function **st** X ∈ dom (.f) **holds** (.f).X = f.X.

Theorem FUNCT_3:9. **for** f **being** Function **holds** (.f).∅ = ∅.

Theorem FUNCT_3:10. **for** f **being** Function **holds** rng (.f) ⊆ bool rng f.

Theorem FUNCT_3:11. **for** f **being** Function **holds** Y ∈ (.f).A **iff ex** X **st** X ∈ dom (.f) & X ∈ A & Y = (.f).X.

Theorem FUNCT_3:12. **for** f **being** Function **holds** (.f).A ⊆ bool rng f.

Theorem FUNCT_3:13. **for** f **being** Function **holds** $(.f)^{-1}$B ⊆ bool dom f.

Theorem FUNCT_3:14. **for** f **being** Function **of** X, D **holds** $(.f)^{-1}$B ⊆ bool X.

Theorem FUNCT_3:15. **for** f **being** Function **holds** ⋃((.f).A) ⊆ f.(⋃A).

Theorem FUNCT_3:16. **for** f **being** Function **st** A ⊆ bool dom f **holds** f.(⋃A) = ⋃((.f).A).

Theorem FUNCT_3:17. **for** f **being** Function **of** X, D **st** A ⊆ bool X **holds** f.(⋃A) = ⋃((.f).A).

Theorem FUNCT_3:18. **for** f **being** Function **holds** ⋃($(.f)^{-1}$B) ⊆ f^{-1}(⋃B).

Theorem FUNCT_3:19. **for** f **being** Function **st** B ⊆ bool rng f **holds** f^{-1}(⋃B) = ⋃($(.f)^{-1}$B).

Theorem FUNCT_3:20. **for** f, g **being** Function **holds.**(g·f) =.g·.f.

Theorem FUNCT_3:21. **for** f **being** Function **holds.**f **is** Function **of** bool dom f, bool rng f.

Theorem FUNCT_3:22. **for** f **being** Function **of** X, Y **st** $Y = \emptyset$ **implies** $X = \emptyset$ **holds.**f **is** Function **of** bool X, bool Y.

Definition

let X, D.

let f **be** Function **of** X, D.

redefine

func.f $\to$ Function **of** bool X, bool D.

Definition

let f **be** Function.

func^{-1}f $\to$ Function **means** dom **it** = bool rng f & **for** Y **st** $Y \in$ bool rng f **holds** **it**.Y = $f^{-1}Y$.

Theorem FUNCT_3:23. **for** g, f **being** Function **holds** g $=^{-1}$f **iff** dom g = bool rng f & **for** Y **st** $Y \in$ bool rng f **holds** g.Y = $f^{-1}Y$.

Theorem FUNCT_3:24. **for** f **being** Function **st** $Y \in$ dom ($^{-1}$f) **holds** ($^{-1}$f).Y = $f^{-1}Y$.

Theorem FUNCT_3:25. **for** f **being** Function **holds** rng ($^{-1}$f) $\subseteq$ bool dom f.

Theorem FUNCT_3:26. **for** f **being** Function **holds** $X \in$ ($^{-1}$f)**.**A **iff** **ex** Y **st** $Y \in$ dom ($^{-1}$f) & $Y \in A$ & X = ($^{-1}$f).Y.

Theorem FUNCT_3:27. **for** f **being** Function **holds** ($^{-1}$f)**.**B $\subseteq$ bool dom f.

Theorem FUNCT_3:28. **for** f **being** Function **holds** $(^{-1}f)^{-1}A \subseteq$ bool rng f.

Theorem FUNCT_3:29. **for** f **being** Function **holds** $\bigcup((^{-1}f).B) \subseteq f^{-1}(\bigcup B)$.

Theorem FUNCT_3:30. **for** f **being** Function **st** $B \subseteq$ bool rng f **holds** $\bigcup((^{-1}f).B) = f^{-1}(\bigcup B)$.

Theorem FUNCT_3:31. **for** f **being** Function **holds** $\bigcup((^{-1}f)^{-1}A) \subseteq$ f**.**($\bigcup$A).

Theorem FUNCT_3:32. **for** f **being** Function **st** $A \subseteq$ bool dom f & f is 1-1 **holds** $\bigcup((^{-1}f)^{-1}A)$ = f**.**($\bigcup$A).

Theorem FUNCT_3:33. **for** f **being** Function **holds** ($^{-1}$f)**.**B $\subseteq$ $(.f)^{-1}B$.

Theorem FUNCT_3:34. **for** f **being** Function **st** f is 1-1 **holds** ($^{-1}$f)**.**B = $(.f)^{-1}B$.

Theorem FUNCT_3:35. **for** f **being** Function, A **be** set **st** $A \subseteq$ bool dom f **holds** $(^{-1}f)^{-1}A \subseteq$ (.f)**.**A.

Theorem FUNCT_3:36. **for** f **being** Function, A **be** set **st** f is 1-1 **holds** (.f)**.**A $\subseteq$ $(^{-1}f)^{-1}A$.

Theorem FUNCT_3:37. **for** f **being** Function, A **be** set **st** f is 1-1 & $A \subseteq$ bool dom f **holds** $(^{-1}f)^{-1}A$ = (.f)**.**A.

Theorem FUNCT_3:38. **for** f, g **being** Function **st** g is 1-1 **holds**$^{-1}$(g·f) $=^{-1}$f·$^{-1}$g.

Theorem FUNCT_3:39. **for** f **being** Function **holds** $^{-1}$f **is** Function **of** bool rng f, bool dom f.

Definition

let A, X.

func χ(A, X) $\rightarrow$ Function **means** dom **it** = X & **for** x **st** x $\in$ X **holds** (x $\in$ A **implies** **it**.x = 1) & (**not** x $\in$ A **implies** **it**.x = 0).

Theorem FUNCT_3:40. **for** f **being** Function **holds** f = χ(A, X) **iff** dom f = X & **for** x **st** x $\in$ X **holds** (x $\in$ A **implies** f.x = 1) & (**not** x $\in$ A **implies** f.x = 0).

Theorem FUNCT_3:41. A $\subseteq$ X & x $\in$ A **implies** χ(A, X).x = 1.

Theorem FUNCT_3:42. x $\in$ X & χ(A, X).x = 1 **implies** x $\in$ A.

Theorem FUNCT_3:43. x $\in$ X$\setminus$A **implies** χ(A, X).x = 0.

Theorem FUNCT_3:44. x $\in$ X & χ(A, X).x = 0 **implies** **not** x $\in$ A.

Theorem FUNCT_3:45. x $\in$ X **implies** χ($\emptyset$, X).x = 0.

Theorem FUNCT_3:46. x $\in$ X **implies** χ(X, X).x = 1.

Theorem FUNCT_3:47. A $\subseteq$ X & B $\subseteq$ X & χ(A, X) = χ(B, X) **implies** A = B.

Theorem FUNCT_3:48. rng χ(A, X) $\subseteq$ {0, 1}.

Theorem FUNCT_3:49. **for** f **being** Function **of** X, {0, 1} **holds** f = χ(f^{-1}{1}, X).

Definition

let A, X.

redefine

func χ(A, X) $\rightarrow$ Function **of** X, {0, 1}.

Theorem FUNCT_3:50. **for** d **being** Element **of** D **holds** χ(A, D).d = 1 **iff** d $\in$ A.

Theorem FUNCT_3:51. **for** d **being** Element **of** D **holds** χ(A, D).d = 0 **iff** **not** d $\in$ A.

Definition

let Y.

let A **be** Subset **of** Y.

func incl (A) $\rightarrow$ Function **of** A, Y **means** **it** = Id A.

Theorem FUNCT_3:52. **for** A **being** Subset **of** Y **holds** incl A = Id A.

Theorem FUNCT_3:53. **for** A **being** Subset **of** Y **holds** incl A = (Id Y)$\restriction$A.

Theorem FUNCT_3:54. **for** A **being** Subset **of** Y **holds** dom incl A = A & rng incl A = A.

Theorem FUNCT_3:55. **for** A **being** Subset **of** Y **st** x $\in$ A **holds** (incl A).x = x.

Theorem FUNCT_3:56. **for** A **being** Subset **of** Y **st** x $\in$ A **holds** incl (A).x $\in$ Y.

Definition

let X, Y.

func π_1(X, Y) → Function **means** dom **it** = [:X, Y:] & **for** x, y **st** x ∈ X & y ∈ Y **holds it**.[x, y] = x.

func π_2(X, Y) → Function **means** dom **it** = [:X, Y:] & **for** x, y **st** x ∈ X & y ∈ Y **holds it**.[x, y] = y.

Theorem FUNCT_3:57. **for** f **being** Function **holds** f = π_1(X, Y) **iff** dom f = [:X, Y:] & **for** x, y **st** x ∈ X & y ∈ Y **holds** f.[x, y] = x.

Theorem FUNCT_3:58. **for** f **being** Function **holds** f = π_2(X, Y) **iff** dom f = [:X, Y:] & **for** x, y **st** x ∈ X & y ∈ Y **holds** f.[x, y] = y.

Theorem FUNCT_3:59. rng π_1(X, Y) ⊆ X.

Theorem FUNCT_3:60. Y ≠ ∅ **implies** rng π_1(X, Y) = X.

Theorem FUNCT_3:61. rng π_2(X, Y) ⊆ Y.

Theorem FUNCT_3:62. X ≠ ∅ **implies** rng π_2(X, Y) = Y.

Definition

let X, Y.

redefine

func π_1(X, Y) → Function **of** [:X, Y:], X.

func π_2(X, Y) → Function **of** [:X, Y:], Y.

Theorem FUNCT_3:63. **for** d1 **being** Element **of** D1 **for** d2 **being** Element **of** D2 **holds** π_1(D1, D2).[d1, d2] = d1.

Theorem FUNCT_3:64. **for** d1 **being** Element **of** D1 **for** d2 **being** Element **of** D2 **holds** π_2(D1, D2).[d1, d2] = d2.

Definition

let X.

func δ(X) → Function **means** dom **it** = X & **for** x **st** x ∈ X **holds it**.x = [x, x].

Theorem FUNCT_3:65. **for** f **being** Function **holds** f = δX **iff** dom f = X & **for** x **st** x ∈ X **holds** f.x = [x, x].

Theorem FUNCT_3:66. rng δX ⊆ [:X, X:].

Definition

let X.

redefine

func δ(X) → Function **of** X, [:X, X:].

Definition

let f, g **be** Function.

func <:f, g:> → Function **means** dom **it** = dom f∩dom g & **for** x **st** x ∈ dom **it** **holds it**.x = [f.x, g.x].

Theorem FUNCT_3:67. **for** f, g, fg **being** Function **holds** fg = <:f, g:> **iff** dom fg = dom f∩dom g & **for** x **st** x ∈ dom fg **holds** fg.x = [f.x, g.x].

Theorem FUNCT_3:68. **for** f, g **being** Function **st** x $\in$ dom f$\cap$dom g **holds** ⦇f, g⦈.x = [f.x, g.x].

Theorem FUNCT_3:69. **for** f, g **being** Function **st** dom f = X & dom g = X & x $\in$ X **holds** ⦇f, g⦈.x = [f.x, g.x].

Theorem FUNCT_3:70. **for** f, g **being** Function **st** dom f = X & dom g = X **holds** dom ⦇f, g⦈ = X.

Theorem FUNCT_3:71. **for** f, g **being** Function **holds** rng ⦇f, g⦈ $\subseteq$ ⟦rng f, rng g⟧.

Theorem FUNCT_3:72. **for** f, g **being** Function **st** dom f = dom g & rng f $\subseteq$ Y & rng g $\subseteq$ Z **holds** π_1(Y, Z)·⦇f, g⦈ = f & π_2(Y, Z)·⦇f, g⦈ = g.

Theorem FUNCT_3:73. ⦇π_1(X, Y), π_2(X, Y)⦈ = Id ⟦X, Y⟧.

Theorem FUNCT_3:74. **for** f, g, h, k **being** Function **st** dom f = dom g & dom k = dom h & ⦇f, g⦈ = ⦇k, h⦈ **holds** f = k & g = h.

Theorem FUNCT_3:75. **for** f, g, h **being** Function **holds** ⦇f·h, g·h⦈ = ⦇f, g⦈·h.

Theorem FUNCT_3:76. **for** f, g **being** Function **holds** ⦇f, g⦈**.**A $\subseteq$ ⟦f**.**A, g**.**A⟧.

Theorem FUNCT_3:77. **for** f, g **being** Function **holds** ⦇f, g⦈$^{-1}$⟦B, C⟧ = f^{-1}B$\cap$g^{-1}C.

Theorem FUNCT_3:78. **for** f **being** Function **of** X, Y **for** g **being** Function **of** X, Z **st** (Y = $\emptyset$ **implies** X = $\emptyset$) & (Z = $\emptyset$ **implies** X = $\emptyset$) **holds** ⦇f, g⦈ **is** Function **of** X, ⟦Y, Z⟧.

Definition

let X, D1, D2.

let f1 **be** Function **of** X, D1.

let f2 **be** Function **of** X, D2.

redefine

func ⦇f1, f2⦈ $\to$ Function **of** X, ⟦D1, D2⟧.

Theorem FUNCT_3:79. **for** f1 **being** Function **of** C, D1 **for** f2 **being** Function **of** C, D2 **for** c **being** Element **of** C **holds** ⦇f1, f2⦈.c = [f1.c, f2.c].

Theorem FUNCT_3:80. **for** f **being** Function **of** X, Y **for** g **being** Function **of** X, Z **st** (Y = $\emptyset$ **implies** X = $\emptyset$) & (Z = $\emptyset$ **implies** X = $\emptyset$) **holds** rng ⦇f, g⦈ $\subseteq$ ⟦Y, Z⟧.

Theorem FUNCT_3:81. **for** f **being** Function **of** X, Y **for** g **being** Function **of** X, Z **st** (Y = $\emptyset$ **implies** X = $\emptyset$) & (Z = $\emptyset$ **implies** X = $\emptyset$) **holds** π_1(Y, Z)·⦇f, g⦈ = f & π_2(Y, Z)·⦇f, g⦈ = g.

Theorem FUNCT_3:82. **for** f **being** Function **of** X, D1 **for** g **being** Function **of** X, D2 **holds** π_1(D1, D2)·⦇f, g⦈ = f & π_2(D1, D2)·⦇f, g⦈ = g.

Theorem FUNCT_3:83. **for** f1, f2 **being** Function **of** X, Y **for** g1, g2 **being** Function **of** X, Z **st** (Y = $\emptyset$ **implies** X = $\emptyset$) & (Z = $\emptyset$ **implies** X = $\emptyset$) & ⦇f1, g1⦈ = ⦇f2, g2⦈ **holds** f1 = f2 & g1 = g2.

Theorem FUNCT_3:84. **for** f1, f2 **being** Function **of** X, D1 **for** g1, g2 **being** Function **of** X, D2 **st** ⦇f1, g1⦈ = ⦇f2, g2⦈ **holds** f1 = f2 & g1 = g2.

Definition

let f, g **be** Function.

func ⟦f, g⟧ → Function **means** dom **it** = ⟦dom f, dom g⟧ & **for** x, y **st** x ∈ dom f & y ∈ dom g **holds** **it**.[x, y] = [f.x, g.y].

Theorem FUNCT_3:85. **for** f, g, fg **being** Function **holds** fg = ⟦f, g⟧ **iff** dom fg = ⟦dom f, dom g⟧ & **for** x, y **st** x ∈ dom f & y ∈ dom g **holds** fg.[x, y] = [f.x, g.y].

Theorem FUNCT_3:86. **for** f, g **being** Function, x, y **st** [x, y] ∈ ⟦dom f, dom g⟧ **holds** ⟦f, g⟧.[x, y] = [f.x, g.y].

Theorem FUNCT_3:87. **for** f, g **being** Function **holds** ⟦f, g⟧ = ⟦(f·π_1(dom f, dom g), g·π_2(dom f, dom g))⟧.

Theorem FUNCT_3:88. **for** f, g **being** Function **holds** rng ⟦f, g⟧ = ⟦rng f, rng g⟧.

Theorem FUNCT_3:89. **for** f, g **being** Function **st** dom f = X & dom g = X **holds** ⟦(f, g)⟧ = ⟦f, g⟧·(δX).

Theorem FUNCT_3:90. ⟦Id X, Id Y⟧ = Id ⟦X, Y⟧.

Theorem FUNCT_3:91. **for** f, g, h, k **being** Function **holds** ⟦f, h⟧·⟦(g, k)⟧ = ⟦(f·g, h·k)⟧.

Theorem FUNCT_3:92. **for** f, g, h, k **being** Function **holds** ⟦f, h⟧·⟦g, k⟧ = ⟦f·g, h·k⟧.

Theorem FUNCT_3:93. **for** f, g **being** Function **holds** ⟦f, g⟧**.**⟦B, C⟧ = ⟦f**.**B, g**.**C⟧.

Theorem FUNCT_3:94. **for** f, g **being** Function **holds** ⟦f, g⟧$^{-1}$⟦B, C⟧ = ⟦f^{-1}B, g^{-1}C⟧.

Theorem FUNCT_3:95. **for** f **being** Function **of** X, Y **for** g **being** Function **of** V, Z **st** (Y = ∅ **implies** X = ∅) & (Z = ∅ **implies** V = ∅) **holds** ⟦f, g⟧ **is** Function **of** ⟦X, V⟧, ⟦Y, Z⟧.

Definition

let X1, X2, D1, D2.

let f1 **be** Function **of** X1, D1.

let f2 **be** Function **of** X2, D2.

redefine

func ⟦f1, f2⟧ → Function **of** ⟦X1, X2⟧, ⟦D1, D2⟧.

Theorem FUNCT_3:96. **for** f1 **being** Function **of** C1, D1 **for** f2 **being** Function **of** C2, D2 **for** c1 **being** Element **of** C1 **for** c2 **being** Element **of** C2 **holds** ⟦f1, f2⟧.[c1, c2] = [f1.c1, f2.c2].

Theorem FUNCT_3:97. **for** f1 **being** Function **of** X1, Y1 **for** f2 **being** Function **of** X2, Y2 **st** (Y1 = ∅ **implies** X1 = ∅) & (Y2 = ∅ **implies** X2 = ∅) **holds** ⟦f1, f2⟧ = ⟦(f1·π_1(X1, X2), f2·π_2(X1, X2))⟧.

Theorem FUNCT_3:98. **for** f1 **being** Function **of** X1, D1 **for** f2 **being** Function **of** X2, D2 **holds** ⟦f1, f2⟧ = ⟦(f1·π_1(X1, X2), f2·π_2(X1, X2))⟧.

Theorem FUNCT_3:99. **for** f1 **being** Function **of** X, Y1 **for** f2 **being** Function **of** X, Y2 **st** (Y1 = ∅ **implies** X = ∅) & (Y2 = ∅ **implies** X = ∅) **holds** ⟦(f1, f2)⟧ = ⟦f1, f2⟧·(δX).

Theorem FUNCT_3:100. **for** f1 **being** Function **of** X, D1 **for** f2 **being** Function **of** X, D2 **holds** $[\!(\text{f1}, \text{f2})\!] = [\![\text{f1}, \text{f2}]\!] \cdot (\delta \text{X})$.

Chapter 11

BINOP_1

Binary Operations.

by

Czesław Byliński [1]

Warsaw University (Białystok)

Summary. In this paper we define binary and unary operations on domains. We also define the following predicates concerning the operations: is commutative, is associative, is the unity of, and is distributive wrt. A number of schemes useful in justifying the existence of the operations are proved.

The symbols used in this article are introduced in the following vocabularies: BOOLE, BINOP, FUNC, FUNC_REL, and COORD. The terminology and notation used in this article have been introduced in the following articles: TARSKI, BOOLE, FUNCT_1, and FUNCT_2.

Definition

let f **be** Function.

let a, b **be** Any.

func f.(a, b) → Any **means it** = f.[a, b].

Theorem BINOP_1:1. **for** f **being** Function **for** a, b **being** Any **holds** f.(a, b) = f.[a, b].

reserve A, B, C **for** DOMAIN.

[1]Supported by RPBP.III-24.C1.

Definition

let A, B, C.

let f **be** Function **of** ⟦A, B⟧, C.

let a **be** Element **of** A.

let b **be** Element **of** B.

redefine

func f.(a, b) → Element **of** C.

Theorem BINOP_1:2. **for** f1, f2 **being** Function **of** ⟦A, B⟧, C **st for** a **being** Element **of** A **for** b **being** Element **of** B **holds** f1.(a, b) = f2.(a, b) **holds** f1 = f2.

Definition

let A.

mode UnOp **of** A → Function **of** A, A **means not contradiction**.

mode BinOp **of** A → Function **of** ⟦A, A⟧, A **means not contradiction**.

Theorem BINOP_1:3. **for** f **being** Function **of** A, A **holds** f **is** UnOp **of** A.

reserve u, u′ **for** UnOp **of** A.

Theorem BINOP_1:4. **for** f **being** Function **of** ⟦A, A⟧, A **holds** f **is** BinOp **of** A.

scheme UnOpEx{A() → DOMAIN, P[(Element **of** A()), Element **of** A()]}: **ex** u **being** UnOp **of** A() **st for** x **being** Element **of** A() **holds** P[x, u.x] **provided** A1: **for** x **being** Element **of** A() **ex** y **being** Element **of** A() **st** P[x, y] **and** A2: **for** x, y1, y2 **being** Element **of** A() **st** P[x, y1] & P[x, y2] **holds** y1 = y2.

scheme UnOpLambda{A() → DOMAIN, F((Element **of** A())) → Element **of** A()}: **ex** u **being** UnOp **of** A() **st for** x **being** Element **of** A() **holds** u.x = F(x).

reserve o, o′ **for** BinOp **of** A.

reserve a, a1, a2, b, b1, b2, c, e, e1, e2 **for** Element **of** A.

Definition

let A, o, a, b.

redefine

func o.(a, b) → Element **of** A.

scheme BinOpEx{A() → DOMAIN, P[(Element **of** A()), (Element **of** A()), Element **of** A()]}: **ex** o **being** BinOp **of** A() **st for** a, b **being** Element **of** A() **holds** P[a, b, o.(a, b)] **provided** A1: **for** x, y **being** Element **of** A() **ex** z **being** Element **of** A() **st** P[x, y, z] **and** A2: **for** x, y **being** Element **of** A() **for** z1, z2 **being** Element **of** A() **st** P[x, y, z1] & P[x, y, z2] **holds** z1 = z2.

scheme BinOpLambda{A() → DOMAIN, O((Element **of** A()), Element **of** A()) → Element **of** A()}: **ex** o **being** BinOp **of** A() **st for** a, b **being** Element **of** A() **holds** o.(a, b) = O(a, b).

Definition

let A, o.

pred o is commutative **means for** a, b **holds** o.(a, b) = o.(b, a).

pred o is associative **means for** a, b, c **holds** o.(a, o.(b, c)) = o.(o.(a, b), c).

pred o is an idempotent **means for** a **holds** o.(a, a) = a.

Theorem BINOP_1:5. o is commutative **iff for** a, b **holds** o.(a, b) = o.(b, a).

Theorem BINOP_1:6. o is associative **iff for** a, b, c **holds** o.(a, o.(b, c)) = o.(o.(a, b), c).

Theorem BINOP_1:7. o is an idempotent **iff for** a **holds** o.(a, a) = a.

Definition

let A, e, o.

pred e is a left unity wrt o **means for** a **holds** o.(e, a) = a.

pred e is a right unity wrt o **means for** a **holds** o.(a, e) = a.

Definition

let A, e, o.

pred e is a unity wrt o **means** e is a left unity wrt o & e is a right unity wrt o.

Theorem BINOP_1:8. e is a left unity wrt o **iff for** a **holds** o.(e, a) = a.

Theorem BINOP_1:9. e is a right unity wrt o **iff for** a **holds** o.(a, e) = a.

Theorem BINOP_1:10. e is a unity wrt o **iff** e is a left unity wrt o & e is a right unity wrt o.

Theorem BINOP_1:11. e is a unity wrt o **iff for** a **holds** o.(e, a) = a & o.(a, e) = a.

Theorem BINOP_1:12. o is commutative **implies** (e is a unity wrt o **iff for** a **holds** o.(e, a) = a).

Theorem BINOP_1:13. o is commutative **implies** (e is a unity wrt o **iff for** a **holds** o.(a, e) = a).

Theorem BINOP_1:14. o is commutative **implies** (e is a unity wrt o **iff** e is a left unity wrt o).

Theorem BINOP_1:15. o is commutative **implies** (e is a unity wrt o **iff** e is a right unity wrt o).

Theorem BINOP_1:16. o is commutative **implies** (e is a left unity wrt o **iff** e is a right unity wrt o).

Theorem BINOP_1:17. e1 is a left unity wrt o & e2 is a right unity wrt o **implies** e1 = e2.

Theorem BINOP_1:18. e1 is a unity wrt o & e2 is a unity wrt o **implies** e1 = e2.

Definition

let A, o.

assume ex e **st** e is a unity wrt o.

func the unity wrt o → Element **of** A **means it** is a unity wrt o.

Theorem BINOP_1:19. (**ex** e **st** e is a unity wrt o) **implies for** e **holds** e = the unity wrt o **iff** e is a unity wrt o.

Definition

let A, o′, o.

pred o′ is left distributive wrt o **means for** a, b, c **holds** o′.(a, o.(b, c)) = o.(o′.(a, b), o′.(a, c)).

pred o′ is right distributive wrt o **means for** a, b, c **holds** o′.(o.(a, b), c) = o.(o′.(a, c), o′.(b, c)).

Definition

let A, o′, o.

pred o′ is distributive wrt o **means** o′ is left distributive wrt o & o′ is right distributive wrt o.

Theorem BINOP_1:20. o′ is left distributive wrt o **iff for** a, b, c **holds** o′.(a, o.(b, c)) = o.(o′.(a, b), o′.(a, c)).

Theorem BINOP_1:21. o′ is right distributive wrt o **iff for** a, b, c **holds** o′.(o.(a, b), c) = o.(o′.(a, c), o′.(b, c)).

Theorem BINOP_1:22. o′ is distributive wrt o **iff** o′ is left distributive wrt o & o′ is right distributive wrt o.

Theorem BINOP_1:23. o′ is distributive wrt o **iff for** a, b, c **holds** o′.(a, o.(b, c)) = o.(o′.(a, b), o′.(a, c)) & o′.(o.(a, b), c) = o.(o′.(a, c), o′.(b, c)).

Theorem BINOP_1:24. o′ is commutative **implies** (o′ is distributive wrt o **iff for** a, b, c **holds** o′.(a, o.(b, c)) = o.(o′.(a, b), o′.(a, c))).

Theorem BINOP_1:25. o′ is commutative **implies** (o′ is distributive wrt o **iff for** a, b, c **holds** o′.(o.(a, b), c) = o.(o′.(a, c), o′.(b, c))).

Theorem BINOP_1:26. o′ is commutative **implies** (o′ is distributive wrt o **iff** o′ is left distributive wrt o).

Theorem BINOP_1:27. o′ is commutative **implies** (o′ is distributive wrt o **iff** o′ is right distributive wrt o).

Theorem BINOP_1:28. o′ is commutative **implies** (o′ is right distributive wrt o **iff** o′ is left distributive wrt o).

Definition

let A, u, o.

pred u is distributive wrt o **means for** a, b **holds** u.(o.(a, b)) = o.((u.a), (u.b)).

Theorem BINOP_1:29. u is distributive wrt o **iff for** a, b **holds** u.(o.(a, b)) = o.((u.a), (u.b)).

Chapter 12

RELAT_1

Relations and Their Basic Properties

by

Edmund Woronowicz [1]

Warsaw University (Białystok)

Summary. We define here: mode Relation as a set of pairs, the domain, the codomain, and the field of relation; the empty and the identity relations, the composition of relations, the image and the inverse image of a set under a relation. Two predicates, = and ⊆, and three functions, ∩, ∪,and ∖ are redefined. Basic facts about the above mentioned notions are presented.

The symbols used in this article are introduced in the following vocabularies: FAM_OP, BOOLE, REAL_1, FUNC_REL, and RELATION. The articles TARSKI and BOOLE provide the terminology and notation for this article.

reserve A, B, X, X1, X2, Y, Y1, Y2 **for** set.

reserve a, b, c, d, x, y, z **for** Any.

Definition

mode Relation → set **means** x ∈ **it** **implies** **ex** y, z **st** x = [y, z].

Theorem RELAT_1:1. **for** R **being** set **st** (**for** x **st** x ∈ R **holds** **ex** y, z **st** x = [y, z]) **holds** R **is** Relation.

reserve P, P1, P2, Q, R, S **for** Relation.

[1]Supported by RPBP.III-24.C1.

Theorem RELAT_1:2. x ∈ R **implies** **ex** y, z **st** x = [y, z].

Theorem RELAT_1:3. A ⊆ R **implies** A **is** Relation.

Theorem RELAT_1:4. {[x, y]} **is** Relation.

Theorem RELAT_1:5. {[a, b], [c, d]} **is** Relation.

Theorem RELAT_1:6. ⟦X, Y⟧ **is** Relation.

scheme Rel_Existence{A() → set, B() → set, P[Any, Any]}: **ex** R **being** Relation **st** **for** x, y **holds** [x, y] ∈ R **iff** x ∈ A() & y ∈ B() & P[x, y].

Definition

let P, R.

redefine

pred P = R **means** **for** a, b **holds** [a, b] ∈ P **iff** [a, b] ∈ R.

Theorem RELAT_1:7. P = R **iff** **for** a, b **holds** [a, b] ∈ P **iff** [a, b] ∈ R.

Definition

let P, R.

redefine

func P∩R → Relation.

func P∪R → Relation.

func P∖R → Relation.

pred P ⊆ R **means** **for** a, b **holds** [a, b] ∈ P **implies** [a, b] ∈ R.

Theorem RELAT_1:8. P ⊆ R **iff** **for** a, b **holds** [a, b] ∈ P **implies** [a, b] ∈ R.

Theorem RELAT_1:9. X∩R **is** Relation & R∩X **is** Relation.

Theorem RELAT_1:10. R∖X **is** Relation.

Definition

let R.

func dom R → set **means** x ∈ **it** **iff** **ex** y **st** [x, y] ∈ R.

Theorem RELAT_1:11. X = dom R **iff** **for** x **holds** x ∈ X **iff** **ex** y **st** [x, y] ∈ R.

Theorem RELAT_1:12. x ∈ dom R **iff** **ex** y **st** [x, y] ∈ R.

Theorem RELAT_1:13. dom (P∪R) = dom P∪dom R.

Theorem RELAT_1:14. dom (P∩R) ⊆ dom P∩dom R.

Theorem RELAT_1:15. dom P∖dom R ⊆ dom (P∖R).

Definition

let R.

func rng R → set **means** y ∈ **it** **iff** **ex** x **st** [x, y] ∈ R.

Theorem RELAT_1:16. X = rng R **iff** **for** x **holds** x ∈ X **iff** **ex** y **st** [y, x] ∈ R.

Theorem RELAT_1:17. x ∈ rng R **iff** **ex** y **st** [y, x] ∈ R.

Theorem RELAT_1:18. x ∈ dom R **implies** **ex** y **st** y ∈ rng R.

Theorem RELAT_1:19. y ∈ rng R **implies** **ex** x **st** x ∈ dom R.

Theorem RELAT_1:20. [x, y] ∈ R **implies** x ∈ dom R & y ∈ rng R.

Theorem RELAT_1:21. R ⊆ ⟦dom R, rng R⟧.

Theorem RELAT_1:22. R∩⟦dom R, rng R⟧ = R.

Theorem RELAT_1:23. R = {[x, y]} **implies** dom R = {x} & rng R = {y}.

Theorem RELAT_1:24. R = {[a, b], [x, y]} **implies** dom R = {a, x} & rng R = {b, y}.

Theorem RELAT_1:25. P ⊆ R **implies** dom P ⊆ dom R & rng P ⊆ rng R.

Theorem RELAT_1:26. rng (P∪R) = rng P∪rng R.

Theorem RELAT_1:27. rng (P∩R) ⊆ rng P∩rng R.

Theorem RELAT_1:28. rng P∖rng R ⊆ rng (P∖R).

Definition

let R.

func field R → set **means** **it** = dom R∪rng R.

Theorem RELAT_1:29. field R = dom R∪rng R.

Theorem RELAT_1:30. [a, b] ∈ R **implies** a ∈ field R & b ∈ field R.

Theorem RELAT_1:31. P ⊆ R **implies** field P ⊆ field R.

Theorem RELAT_1:32. R = {[x, y]} **implies** field R = {x, y}.

Theorem RELAT_1:33. field (P∪R) = field P∪field R.

Theorem RELAT_1:34. field (P∩R) ⊆ field P∩field R.

Definition

let R.

func $R^{\smile}$ → Relation **means** [x, y] ∈ **it** **iff** [y, x] ∈ R.

Theorem RELAT_1:35. R = $P^{\smile}$ **iff** **for** x, y **holds** [x, y] ∈ R **iff** [y, x] ∈ P.

Theorem RELAT_1:36. [x, y] ∈ $P^{\smile}$ **iff** [y, x] ∈ P.

Theorem RELAT_1:37. $(R^{\smile})^{\smile}$ = R.

Theorem RELAT_1:38. field R = field ($R^{\smile}$).

Theorem RELAT_1:39. $(P \cap R)^{\smile} = P^{\smile} \cap R^{\smile}$.

Theorem RELAT_1:40. $(P \cup R)^{\smile} = P^{\smile} \cup R^{\smile}$.

Theorem RELAT_1:41. $(P \setminus R)^{\smile} = P^{\smile} \setminus R^{\smile}$.

Definition

let P, R.

func P·R → Relation **means** [x, y] ∈ **it** **iff** **ex** z **st** [x, z] ∈ P & [z, y] ∈ R.

Theorem RELAT_1:42. $Q = P\cdot R$ **iff for** x, y **holds** $[x, y] \in Q$ **iff ex** z **st** $[x, z] \in P$ & $[z, y] \in R$.

Theorem RELAT_1:43. $[x, y] \in P\cdot R$ **iff ex** z **st** $[x, z] \in P$ & $[z, y] \in R$.

Theorem RELAT_1:44. $\mathsf{dom}\ (P\cdot R) \subseteq \mathsf{dom}\ P$.

Theorem RELAT_1:45. $\mathsf{rng}\ (P\cdot R) \subseteq \mathsf{rng}\ R$.

Theorem RELAT_1:46. $\mathsf{rng}\ R \subseteq \mathsf{dom}\ P$ **implies** $\mathsf{dom}\ (R\cdot P) = \mathsf{dom}\ R$.

Theorem RELAT_1:47. $\mathsf{dom}\ P \subseteq \mathsf{rng}\ R$ **implies** $\mathsf{rng}\ (R\cdot P) = \mathsf{rng}\ P$.

Theorem RELAT_1:48. $P \subseteq R$ **implies** $Q\cdot P \subseteq Q\cdot R$.

Theorem RELAT_1:49. $P \subseteq Q$ **implies** $P\cdot R \subseteq Q\cdot R$.

Theorem RELAT_1:50. $P \subseteq R$ & $Q \subseteq S$ **implies** $P\cdot Q \subseteq R\cdot S$.

Theorem RELAT_1:51. $P\cdot(R\cup Q) = (P\cdot R)\cup(P\cdot Q)$.

Theorem RELAT_1:52. $P\cdot(R\cap Q) \subseteq (P\cdot R)\cap(P\cdot Q)$.

Theorem RELAT_1:53. $(P\cdot R)\setminus(P\cdot Q) \subseteq P\cdot(R\setminus Q)$.

Theorem RELAT_1:54. $(P\cdot R)^{\smile} = R^{\smile}\cdot P^{\smile}$.

Theorem RELAT_1:55. $(P\cdot R)\cdot Q = P\cdot(R\cdot Q)$.

Definition

func $\emptyset \to$ Relation **means not** $[x, y] \in$ **it**.

Theorem RELAT_1:56. $R = \emptyset$**iff for** x, y **holds not** $[x, y] \in R$.

Theorem RELAT_1:57. **not** $[x, y] \in \emptyset$.

Theorem RELAT_1:58. $\emptyset\subseteq [\![A, B]\!]$.

Theorem RELAT_1:59. $\emptyset\subseteq R$.

Theorem RELAT_1:60. $\mathsf{dom}\ \emptyset= \emptyset$ & $\mathsf{rng}\ \emptyset= \emptyset$.

Theorem RELAT_1:61. $\emptyset\cap R = \emptyset$& $\emptyset\cup R = R$.

Theorem RELAT_1:62. $\emptyset\cdot R = \emptyset$& $R\cdot\emptyset= \emptyset$.

Theorem RELAT_1:63. $R\cdot\emptyset= \emptyset\cdot R$.

Theorem RELAT_1:64. $\mathsf{dom}\ R = \emptyset$ **or** $\mathsf{rng}\ R = \emptyset$ **implies** $R = \emptyset$.

Theorem RELAT_1:65. $\mathsf{dom}\ R = \emptyset$ **iff** $\mathsf{rng}\ R = \emptyset$.

Theorem RELAT_1:66. $\emptyset^{\smile} = \emptyset$.

Theorem RELAT_1:67. $\mathsf{rng}\ R\cap\mathsf{dom}\ P = \emptyset$ **implies** $R\cdot P = \emptyset$.

Definition

let X.

func $\triangle X \to$ Relation **means** $[x, y] \in$ **it iff** $x \in X$ & $x = y$.

Theorem RELAT_1:68. $P = \triangle X$ **iff for** x, y **holds** $[x, y] \in P$ **iff** $x \in X$ & $x = y$.

Theorem RELAT_1:69. $[x, y] \in \triangle X$ **iff** $x \in X$ & $x = y$.

Theorem RELAT_1:70. $x \in X$ **iff** $[x, x] \in \triangle X$.

Theorem RELAT_1:71. dom △X = X & rng △X = X.

Theorem RELAT_1:72. (△X)⌣ = △X.

Theorem RELAT_1:73. (**for** x **st** x ∈ X **holds** [x, x] ∈ R) **implies** △X ⊆ R.

Theorem RELAT_1:74. [x, y] ∈ (△X)·R **iff** x ∈ X & [x, y] ∈ R.

Theorem RELAT_1:75. [x, y] ∈ R·△Y **iff** y ∈ Y & [x, y] ∈ R.

Theorem RELAT_1:76. R·(△X) ⊆ R & (△X)·R ⊆ R.

Theorem RELAT_1:77. dom R ⊆ X **implies** (△X)·R = R.

Theorem RELAT_1:78. (△dom R)·R = R.

Theorem RELAT_1:79. rng R ⊆ Y **implies** R·(△Y) = R.

Theorem RELAT_1:80. R·(△rng R) = R.

Theorem RELAT_1:81. △∅ = ∅.

Theorem RELAT_1:82. dom R = X & rng P2 ⊆ X & P2·R = △(dom P1) & R·P1 = △X **implies** P1 = P2.

Theorem RELAT_1:83. dom R = X & rng P2 = X & P2·R = △(dom P1) & R·P1 = △X **implies** P1 = P2.

Definition

let R, X.

func R↾X → Relation **means** [x, y] ∈ **it** **iff** x ∈ X & [x, y] ∈ R.

Theorem RELAT_1:84. P = R↾X **iff for** x, y **holds** [x, y] ∈ P **iff** x ∈ X & [x, y] ∈ R.

Theorem RELAT_1:85. [x, y] ∈ R↾X **iff** x ∈ X & [x, y] ∈ R.

Theorem RELAT_1:86. x ∈ dom (R↾X) **iff** x ∈ X & x ∈ dom R.

Theorem RELAT_1:87. dom (R↾X) ⊆ X.

Theorem RELAT_1:88. R↾X ⊆ R.

Theorem RELAT_1:89. dom (R↾X) ⊆ dom R.

Theorem RELAT_1:90. dom (R↾X) = dom R∩X.

Theorem RELAT_1:91. X ⊆ dom R **implies** dom (R↾X) = X.

Theorem RELAT_1:92. (R↾X)·P ⊆ R·P.

Theorem RELAT_1:93. P·(R↾X) ⊆ P·R.

Theorem RELAT_1:94. R↾X = (△X)·R.

Theorem RELAT_1:95. R↾X = ∅**iff** (dom R)∩X = ∅.

Theorem RELAT_1:96. R↾X = R∩⟦X, rng R⟧.

Theorem RELAT_1:97. dom R ⊆ X **implies** R↾X = R.

Theorem RELAT_1:98. R↾dom R = R.

Theorem RELAT_1:99. rng (R↾X) ⊆ rng R.

Theorem RELAT_1:100. (R↾X)↾Y = R↾(X∩Y).

Theorem RELAT_1:101. $(R \restriction X) \restriction X = R \restriction X$.

Theorem RELAT_1:102. $X \subseteq Y$ **implies** $(R \restriction X) \restriction Y = R \restriction X$.

Theorem RELAT_1:103. $Y \subseteq X$ **implies** $(R \restriction X) \restriction Y = R \restriction Y$.

Theorem RELAT_1:104. $X \subseteq Y$ **implies** $R \restriction X \subseteq R \restriction Y$.

Theorem RELAT_1:105. $P \subseteq R$ **implies** $P \restriction X \subseteq R \restriction X$.

Theorem RELAT_1:106. $P \subseteq R$ & $X \subseteq Y$ **implies** $P \restriction X \subseteq R \restriction Y$.

Theorem RELAT_1:107. $R \restriction (X \cup Y) = (R \restriction X) \cup (R \restriction Y)$.

Theorem RELAT_1:108. $R \restriction (X \cap Y) = (R \restriction X) \cap (R \restriction Y)$.

Theorem RELAT_1:109. $R \restriction (X \setminus Y) = R \restriction X \setminus R \restriction Y$.

Theorem RELAT_1:110. $R \restriction \emptyset = \emptyset$.

Theorem RELAT_1:111. $\emptyset \restriction X = \emptyset$.

Theorem RELAT_1:112. $(P \cdot R) \restriction X = (P \restriction X) \cdot R$.

Definition

let Y, R.

func $Y \restriction R \to$ Relation **means** $[x, y] \in$ **it iff** $y \in Y$ & $[x, y] \in R$.

Theorem RELAT_1:113. $P = Y \restriction R$ **iff for** x, y **holds** $[x, y] \in P$ **iff** $y \in Y$ & $[x, y] \in R$.

Theorem RELAT_1:114. $[x, y] \in Y \restriction R$ **iff** $y \in Y$ & $[x, y] \in R$.

Theorem RELAT_1:115. $y \in \mathsf{rng}\ (Y \restriction R)$ **iff** $y \in Y$ & $y \in \mathsf{rng}\ R$.

Theorem RELAT_1:116. $\mathsf{rng}\ (Y \restriction R) \subseteq Y$.

Theorem RELAT_1:117. $Y \restriction R \subseteq R$.

Theorem RELAT_1:118. $\mathsf{rng}\ (Y \restriction R) \subseteq \mathsf{rng}\ R$.

Theorem RELAT_1:119. $\mathsf{rng}\ (Y \restriction R) = \mathsf{rng}\ R \cap Y$.

Theorem RELAT_1:120. $Y \subseteq \mathsf{rng}\ R$ **implies** $\mathsf{rng}\ (Y \restriction R) = Y$.

Theorem RELAT_1:121. $(Y \restriction R) \cdot P \subseteq R \cdot P$.

Theorem RELAT_1:122. $P \cdot (Y \restriction R) \subseteq P \cdot R$.

Theorem RELAT_1:123. $Y \restriction R = R \cdot (\triangle Y)$.

Theorem RELAT_1:124. $Y \restriction R = R \cap [\![\mathsf{dom}\ R, Y]\!]$.

Theorem RELAT_1:125. $\mathsf{rng}\ R \subseteq Y$ **implies** $Y \restriction R = R$.

Theorem RELAT_1:126. $\mathsf{rng}\ R \restriction R = R$.

Theorem RELAT_1:127. $Y \restriction (X \restriction R) = (Y \cap X) \restriction R$.

Theorem RELAT_1:128. $Y \restriction (Y \restriction R) = Y \restriction R$.

Theorem RELAT_1:129. $X \subseteq Y$ **implies** $Y \restriction (X \restriction R) = X \restriction R$.

Theorem RELAT_1:130. $Y \subseteq X$ **implies** $Y \restriction (X \restriction R) = Y \restriction R$.

Theorem RELAT_1:131. $X \subseteq Y$ **implies** $X \restriction R \subseteq Y \restriction R$.

Theorem RELAT_1:132. $P1 \subseteq P2$ **implies** $Y \restriction P1 \subseteq Y \restriction P2$.

Theorem RELAT_1:133. P1 ⊆ P2 & Y1 ⊆ Y2 **implies** Y1↾P1 ⊆ Y2↾P2.

Theorem RELAT_1:134. (X∪Y)↾R = (X↾R)∪(Y↾R).

Theorem RELAT_1:135. (X∩Y)↾R = X↾R∩Y↾R.

Theorem RELAT_1:136. (X∖Y)↾R = X↾R∖Y↾R.

Theorem RELAT_1:137. ∅↾R = ∅.

Theorem RELAT_1:138. Y↾∅= ∅.

Theorem RELAT_1:139. Y↾(P·R) = P·(Y↾R).

Theorem RELAT_1:140. (Y↾R)↾X = Y↾(R↾X).

Definition

let R, X.

func R.X → set **means** y ∈ **it** **iff** **ex** x **st** [x, y] ∈ R & x ∈ X.

Theorem RELAT_1:141. Y = R.X **iff** **for** y **holds** y ∈ Y **iff** **ex** x **st** [x, y] ∈ R & x ∈ X.

Theorem RELAT_1:142. y ∈ R.X **iff** **ex** x **st** [x, y] ∈ R & x ∈ X.

Theorem RELAT_1:143. y ∈ R.X **iff** **ex** x **st** x ∈ dom R & [x, y] ∈ R & x ∈ X.

Theorem RELAT_1:144. R.X ⊆ rng R.

Theorem RELAT_1:145. R.X = R.(dom R∩X).

Theorem RELAT_1:146. R.dom R = rng R.

Theorem RELAT_1:147. R.X ⊆ R.(dom R).

Theorem RELAT_1:148. rng (R↾X) = R.X.

Theorem RELAT_1:149. R.∅ = ∅.

Theorem RELAT_1:150. ∅.X = ∅.

Theorem RELAT_1:151. R.X = ∅ **iff** dom R∩X = ∅.

Theorem RELAT_1:152. X ≠ ∅ & X ⊆ dom R **implies** R.X ≠ ∅.

Theorem RELAT_1:153. R.(X∪Y) = R.X∪R.Y.

Theorem RELAT_1:154. R.(X∩Y) ⊆ R.X∩R.Y.

Theorem RELAT_1:155. R.X∖R.Y ⊆ R.(X∖Y).

Theorem RELAT_1:156. X ⊆ Y **implies** R.X ⊆ R.Y.

Theorem RELAT_1:157. P ⊆ R **implies** P.X ⊆ R.X.

Theorem RELAT_1:158. P ⊆ R & X ⊆ Y **implies** P.X ⊆ R.Y.

Theorem RELAT_1:159. (P·R).X = R.(P.X).

Theorem RELAT_1:160. rng (P·R) = R.(rng P).

Theorem RELAT_1:161. (R↾X).Y ⊆ R.Y.

Theorem RELAT_1:162. R↾X = ∅**iff** (dom R)∩X = ∅.

Theorem RELAT_1:163. (dom R)∩X ⊆ (R$^{\smile}$).(R.X).

Definition

let R, Y.

func $R^{-1}Y \rightarrow$ set **means** x $\in$ **it** **iff** **ex** y **st** [x, y] $\in$ R & y $\in$ Y.

Theorem RELAT_1:164. X = $R^{-1}Y$ **iff** **for** x **holds** x $\in$ X **iff** **ex** y **st** [x, y] $\in$ R & y $\in$ Y.

Theorem RELAT_1:165. x $\in R^{-1}Y$ **iff** **ex** y **st** [x, y] $\in$ R & y $\in$ Y.

Theorem RELAT_1:166. x $\in R^{-1}Y$ **iff** **ex** y **st** y $\in$ rng R & [x, y] $\in$ R & y $\in$ Y.

Theorem RELAT_1:167. $R^{-1}Y \subseteq$ dom R.

Theorem RELAT_1:168. $R^{-1}Y = R^{-1}$(rng R$\cap$Y).

Theorem RELAT_1:169. R^{-1} rng R = dom R.

Theorem RELAT_1:170. $R^{-1}Y \subseteq R^{-1}$ rng R.

Theorem RELAT_1:171. $R^{-1}\emptyset = \emptyset$.

Theorem RELAT_1:172. $\emptyset^{-1}Y = \emptyset$.

Theorem RELAT_1:173. $R^{-1}Y = \emptyset$ **iff** rng R$\cap$Y = $\emptyset$.

Theorem RELAT_1:174. Y $\neq \emptyset$ & Y $\subseteq$ rng R **implies** $R^{-1}Y \neq \emptyset$.

Theorem RELAT_1:175. $R^{-1}(X \cup Y) = R^{-1}X \cup R^{-1}Y$.

Theorem RELAT_1:176. $R^{-1}(X \cap Y) \subseteq R^{-1}Y \cap R^{-1}Y$.

Theorem RELAT_1:177. $R^{-1}X \setminus R^{-1}Y \subseteq R^{-1}(X \setminus Y)$.

Theorem RELAT_1:178. X $\subseteq$ Y **implies** $R^{-1}X \subseteq R^{-1}Y$.

Theorem RELAT_1:179. P $\subseteq$ R **implies** $P^{-1}Y \subseteq R^{-1}Y$.

Theorem RELAT_1:180. P $\subseteq$ R & X $\subseteq$ Y **implies** $P^{-1}X \subseteq R^{-1}Y$.

Theorem RELAT_1:181. $(P \cdot R)^{-1}Y = P^{-1}(R^{-1}Y)$.

Theorem RELAT_1:182. dom $(P \cdot R) = P^{-1}$(dom R).

Theorem RELAT_1:183. (rng R)$\cap$Y $\subseteq (R^{\smile})^{-1}(R^{-1}Y)$.

Chapter 13

GRFUNC_1

Graphs of Functions.

by

Czesław Byliński [1]

Warsaw University (Białystok)

Summary. The graph of a function is defined in *Functions and their Basic Properties* (FUNCT_1). In this paper the graph of a function is redefined as a Relation. Operations on functions are interpreted as the corresponding operations on relations. Some theorems about graphs of functions are proved.

The symbols used in this article are introduced in the following vocabularies: BOOLE, REAL_1, FUNC_REL, RELATION, and FUNC. The terminology and notation used in this article have been introduced in the following articles: TARSKI, BOOLE, FUNCT_1, and RELAT_1.

reserve X, X1, X2, Y, Y1, Y2, Z, Z1, Z2 **for** set, p, x, x1, x2, y, y1, y2, z, z1, z2 **for** Any.

reserve f, f1, f2, g, g1, g2, h, h1, h2 **for** Function.

Definition

let f.

redefine

func graph f → Relation.

[1]Supported by RPBP.III-24.C1.

Theorem GRFUNC_1:1. **for** R **being** Relation **st** **for** x, y1, y2 **st** [x, y1] ∈ R & [x, y2] ∈ R **holds** y1 = y2 **holds** **ex** f **st** graph f = R.

Theorem GRFUNC_1:2. y ∈ rng f **iff** **ex** x **st** [x, y] ∈ graph f.

Theorem GRFUNC_1:3. dom graph f = dom f & rng graph f = rng f.

Theorem GRFUNC_1:4. graph f ⊆ ⟦dom f, rng f⟧.

Theorem GRFUNC_1:5. (**for** x, y **holds** [x, y] ∈ graph f1 **iff** [x, y] ∈ graph f2) **implies** f1 = f2.

Theorem GRFUNC_1:6. **for** G **being** set **st** G ⊆ graph f **holds** **ex** g **st** graph g = G.

Theorem GRFUNC_1:7. graph f ⊆ graph g **implies** dom f ⊆ dom g & rng f ⊆ rng g.

Theorem GRFUNC_1:8. graph f ⊆ graph g **iff** dom f ⊆ dom g & (**for** x **st** x ∈ dom f **holds** f.x = g.x).

Theorem GRFUNC_1:9. dom f = dom g & graph f ⊆ graph g **implies** f = g.

Theorem GRFUNC_1:10. [x, z] ∈ graph (g·f) **iff** **ex** y **st** [x, y] ∈ graph f & [y, z] ∈ graph g.

Theorem GRFUNC_1:11. (graph f)·(graph g) = graph (g·f).

Theorem GRFUNC_1:12. [x, z] ∈ graph (g·f) **implies** [x, f.x] ∈ graph f & [f.x, z] ∈ graph g.

Theorem GRFUNC_1:13. graph h ⊆ graph f **implies** graph (g·h) ⊆ graph (g·f) & graph (h·g) ⊆ graph (f·g).

Theorem GRFUNC_1:14. graph g2 ⊆ graph g1 & graph f2 ⊆ graph f1 **implies** graph (g2·f2) ⊆ graph (g1·f1).

Theorem GRFUNC_1:15. **ex** f **st** graph f = {[x, y]}.

Theorem GRFUNC_1:16. graph f = {[x, y]} **implies** f.x = y.

Theorem GRFUNC_1:17. graph f = {[x, y]} **implies** dom f = {x} & rng f = {y}.

Theorem GRFUNC_1:18. dom f = {x} **implies** graph f = {[x, f.x]}.

Theorem GRFUNC_1:19. (**ex** f **st** graph f = {[x1, y1], [x2, y2]}) **iff** (x1 = x2 **implies** y1 = y2).

Theorem GRFUNC_1:20. **ex** f **st** graph f = ∅.

Theorem GRFUNC_1:21. graph f = ∅ **implies** dom f = ∅ & rng f = ∅.

Theorem GRFUNC_1:22. rng f = ∅ **or** dom f = ∅ **implies** graph f = ∅.

Theorem GRFUNC_1:23. rng f∩dom g = ∅ **implies** graph (g·f) = ∅.

Theorem GRFUNC_1:24. graph g = ∅ **implies** graph (g·f) = ∅ & graph (f·g) = ∅.

Theorem GRFUNC_1:25. f is 1-1 **iff** **for** x1, x2, y **st** [x1, y] ∈ graph f & [x2, y] ∈ graph f **holds** x1 = x2.

Theorem GRFUNC_1:26. graph g ⊆ graph f & f is 1-1 **implies** g is 1-1.

Theorem GRFUNC_1:27. (**ex** g **st** graph g = graph f∩X) & (**ex** g **st** graph g = X∩graph f).

Theorem GRFUNC_1:28. graph h = graph f$\cap$graph g **implies** dom h $\subseteq$ dom f$\cap$dom g & rng h $\subseteq$ rng f$\cap$rng g.

Theorem GRFUNC_1:29. graph h = graph f$\cap$graph g & x $\in$ dom h **implies** h.x = f.x & h.x = g.x.

Theorem GRFUNC_1:30. (f is 1-1 **or** g is 1-1) & graph h = graph f$\cap$graph g **implies** h is 1-1.

Theorem GRFUNC_1:31. dom f$\cap$dom g = $\emptyset$ **implies ex** h **st** graph h = graph f$\cup$graph g.

Theorem GRFUNC_1:32. graph f $\subseteq$ graph h & graph g $\subseteq$ graph h **implies ex** h1 **st** graph h1 = graph f$\cup$graph g.

Theorem GRFUNC_1:33. graph h = graph (f)$\cup$graph (g) **implies** dom h = dom f$\cup$dom g & rng h = rng f$\cup$rng g.

Theorem GRFUNC_1:34. x $\in$ dom f & graph h = graph f$\cup$graph g **implies** h.x = f.x.

Theorem GRFUNC_1:35. x $\in$ dom g & graph h = graph f$\cup$graph g **implies** h.x = g.x.

Theorem GRFUNC_1:36. x $\in$ dom h & graph h = graph f$\cup$graph g **implies** h.x = f.x **or** h.x = g.x.

Theorem GRFUNC_1:37. f is 1-1 & g is 1-1 & graph h = graph f$\cup$graph g & rng f$\cap$rng g = $\emptyset$ **implies** h is 1-1.

Theorem GRFUNC_1:38. **ex** g **st** graph g = graph (f)$\setminus$X.

Theorem GRFUNC_1:39. [x, y] $\in$ graph Id (X) **iff** x $\in$ X & x = y.

Theorem GRFUNC_1:40. graph Id X = $\triangle$X.

Theorem GRFUNC_1:41. x $\in$ X **iff** [x, x] $\in$ graph Id (X).

Theorem GRFUNC_1:42. [x, y] $\in$ graph (f$\cdot$Id (X)) **iff** x $\in$ X & [x, y] $\in$ graph f.

Theorem GRFUNC_1:43. [x, y] $\in$ graph (Id (Y)$\cdot$f) **iff** [x, y] $\in$ graph f & y $\in$ Y.

Theorem GRFUNC_1:44. graph (f$\cdot$Id (X)) $\subseteq$ graph f & graph (Id (X)$\cdot$f) $\subseteq$ graph (f).

Theorem GRFUNC_1:45. graph Id $\emptyset$ = $\emptyset$.

Theorem GRFUNC_1:46. graph f = $\emptyset$ **implies** f is 1-1.

Theorem GRFUNC_1:47. f is 1-1 **implies for** x, y **holds** [y, x] $\in$ graph (f^{-1}) **iff** [x, y] $\in$ graph f.

Theorem GRFUNC_1:48. f is 1-1 **implies** graph (f^{-1}) = (graph f)$^{\smile}$.

Theorem GRFUNC_1:49. graph f = $\emptyset$ **implies** graph (f^{-1}) = $\emptyset$.

Theorem GRFUNC_1:50. [x, y] $\in$ graph (f$\restriction$X) **iff** x $\in$ X & [x, y] $\in$ graph f.

Theorem GRFUNC_1:51. graph (f$\restriction$X) = (graph f)$\restriction$X.

Theorem GRFUNC_1:52. x $\in$ dom f & x $\in$ X **iff** [x, f.x] $\in$ graph (f$\restriction$X).

Theorem GRFUNC_1:53. graph (f$\restriction$X) $\subseteq$ graph f.

Theorem GRFUNC_1:54. graph ((f$\restriction$X)$\cdot$h) $\subseteq$ graph (f$\cdot$h) & graph (g$\cdot$(f$\restriction$X)) $\subseteq$ graph (g$\cdot$f).

Theorem GRFUNC_1:55. $\mathsf{graph}\ (f{\restriction}X) = \mathsf{graph}\ (f) \cap [\![X, \mathsf{rng}\ f]\!]$.

Theorem GRFUNC_1:56. $X \subseteq Y$ **implies** $\mathsf{graph}\ (f{\restriction}X) \subseteq \mathsf{graph}\ (f{\restriction}Y)$.

Theorem GRFUNC_1:57. $\mathsf{graph}\ f1 \subseteq \mathsf{graph}\ f2$ **implies** $\mathsf{graph}\ (f1{\restriction}X) \subseteq \mathsf{graph}\ (f2{\restriction}X)$.

Theorem GRFUNC_1:58. $\mathsf{graph}\ f1 \subseteq \mathsf{graph}\ f2\ \&\ X1 \subseteq X2$ **implies** $\mathsf{graph}\ (f1{\restriction}X1) \subseteq \mathsf{graph}\ (f2{\restriction}X2)$.

Theorem GRFUNC_1:59. $\mathsf{graph}\ (f{\restriction}(X \cup Y)) = \mathsf{graph}\ (f{\restriction}X) \cup \mathsf{graph}\ (f{\restriction}Y)$.

Theorem GRFUNC_1:60. $\mathsf{graph}\ (f{\restriction}(X \cap Y)) = \mathsf{graph}\ (f{\restriction}X) \cap \mathsf{graph}\ (f{\restriction}Y)$.

Theorem GRFUNC_1:61. $\mathsf{graph}\ (f{\restriction}(X \setminus Y)) = \mathsf{graph}\ (f{\restriction}X) \setminus \mathsf{graph}\ (f{\restriction}Y)$.

Theorem GRFUNC_1:62. $\mathsf{graph}\ (f{\restriction}\emptyset) = \emptyset$.

Theorem GRFUNC_1:63. $\mathsf{graph}\ f = \emptyset$ **implies** $\mathsf{graph}\ (f{\restriction}X) = \emptyset$.

Theorem GRFUNC_1:64. $\mathsf{graph}\ g \subseteq \mathsf{graph}\ f$ **implies** $f{\restriction}\mathsf{dom}\ g = g$.

Theorem GRFUNC_1:65. $[x, y] \in \mathsf{graph}\ (Y{\restriction}f)$ **iff** $y \in Y\ \&\ [x, y] \in \mathsf{graph}\ f$.

Theorem GRFUNC_1:66. $\mathsf{graph}\ (Y{\restriction}f) = Y{\restriction}(\mathsf{graph}\ f)$.

Theorem GRFUNC_1:67. $x \in \mathsf{dom}\ f\ \&\ f.x \in Y$ **iff** $[x, f.x] \in \mathsf{graph}\ (Y{\restriction}f)$.

Theorem GRFUNC_1:68. $\mathsf{graph}\ (Y{\restriction}f) \subseteq \mathsf{graph}\ (f)$.

Theorem GRFUNC_1:69. $\mathsf{graph}\ ((Y{\restriction}f)\cdot h) \subseteq \mathsf{graph}\ (f\cdot h)\ \&\ \mathsf{graph}\ (g\cdot(Y{\restriction}f)) \subseteq \mathsf{graph}\ (g\cdot f)$.

Theorem GRFUNC_1:70. $\mathsf{graph}\ (Y{\restriction}f) = \mathsf{graph}\ (f) \cap [\![\mathsf{dom}\ f, Y]\!]$.

Theorem GRFUNC_1:71. $X \subseteq Y$ **implies** $\mathsf{graph}\ (X{\restriction}f) \subseteq \mathsf{graph}\ (Y{\restriction}f)$.

Theorem GRFUNC_1:72. $\mathsf{graph}\ f1 \subseteq \mathsf{graph}\ f2$ **implies** $\mathsf{graph}\ (Y{\restriction}f1) \subseteq \mathsf{graph}\ (Y{\restriction}f2)$.

Theorem GRFUNC_1:73. $\mathsf{graph}\ f1 \subseteq \mathsf{graph}\ f2\ \&\ Y1 \subseteq Y2$ **implies** $\mathsf{graph}\ (Y1{\restriction}f1) \subseteq \mathsf{graph}\ (Y2{\restriction}f2)$.

Theorem GRFUNC_1:74. $\mathsf{graph}\ ((X \cup Y){\restriction}f) = \mathsf{graph}\ (X{\restriction}f) \cup \mathsf{graph}\ (Y{\restriction}f)$.

Theorem GRFUNC_1:75. $\mathsf{graph}\ ((X \cap Y){\restriction}f) = \mathsf{graph}\ (X{\restriction}f) \cap \mathsf{graph}\ (Y{\restriction}f)$.

Theorem GRFUNC_1:76. $\mathsf{graph}\ ((X \setminus Y){\restriction}f) = \mathsf{graph}\ (X{\restriction}f) \setminus \mathsf{graph}\ (Y{\restriction}f)$.

Theorem GRFUNC_1:77. $\mathsf{graph}\ (\emptyset{\restriction}f) = \emptyset$.

Theorem GRFUNC_1:78. $\mathsf{graph}\ f = \emptyset$ **implies** $\mathsf{graph}\ (Y{\restriction}f) = \emptyset$.

Theorem GRFUNC_1:79. $\mathsf{graph}\ g \subseteq \mathsf{graph}\ f\ \&\ f$ is 1-1 **implies** $\mathsf{rng}\ g{\restriction}f = g$.

Theorem GRFUNC_1:80. $y \in f.X$ **iff** **ex** x **st** $[x, y] \in \mathsf{graph}\ f\ \&\ x \in X$.

Theorem GRFUNC_1:81. $f.X = (\mathsf{graph}\ f).X$.

Theorem GRFUNC_1:82. $\mathsf{graph}\ f = \emptyset$ **implies** $f.X = \emptyset$.

Theorem GRFUNC_1:83. $\mathsf{graph}\ f1 \subseteq \mathsf{graph}\ f2$ **implies** $f1.X \subseteq f2.X$.

Theorem GRFUNC_1:84. $\mathsf{graph}\ f1 \subseteq \mathsf{graph}\ f2\ \&\ X1 \subseteq X2$ **implies** $f1.X1 \subseteq f2.X2$.

Theorem GRFUNC_1:85. $x \in f^{-1}Y$ **iff** **ex** y **st** $[x, y] \in \mathsf{graph}\ f\ \&\ y \in Y$.

Theorem GRFUNC_1:86. $f^{-1}Y = (\mathsf{graph}\ f)^{-1}Y$.

Theorem GRFUNC_1:87. x $\in$ f^{-1}Y **iff** [x, f.x] $\in$ graph f & f.x $\in$ Y.

Theorem GRFUNC_1:88. graph f = $\emptyset$ **implies** f^{-1}Y = $\emptyset$.

Theorem GRFUNC_1:89. graph f1 $\subseteq$ graph f2 **implies** $f1^{-1}$Y $\subseteq$ $f2^{-1}$Y.

Theorem GRFUNC_1:90. graph f1 $\subseteq$ graph f2 & Y1 $\subseteq$ Y2 **implies** $f1^{-1}$Y1 $\subseteq$ $f2^{-1}$Y2.

Chapter 14

RELAT_2

Properties of Binary Relations

by

Edmund Woronowicz [1]

Warsaw University (Białystok)

Anna Zalewska [2]

Warsaw University (Białystok)

Summary. The paper contains definitions of some properties of binary relations: reflexivity, irreflexivity, symmetry, asymmetry, antisymmetry, connectedness, strong connectedness, and transitivity. Basic theorems relating the above mentioned notions are given.

The symbols used in this article are introduced in the following vocabularies: BOOLE, REAL_1, FUNC_REL, RELATION, and REL_REL. The terminology and notation used in this article have been introduced in the following articles: TARSKI, BOOLE, and RELAT_1.

reserve X, Y **for** set.

reserve a, b, c, x, y, z **for** Any.

reserve P, R **for** Relation.

Definition

let R, X.

[1]Supported by RPBP.III-24.C1.

[2]Supported by RPBP.III-24.C1.

pred R is reflexive in X **means** x ∈ X **implies** [x, x] ∈ R.

pred R is irreflexive in X **means** x ∈ X **implies not** [x, x] ∈ R.

pred R is symmetric in X **means** x ∈ X & y ∈ X & [x, y] ∈ R **implies** [y, x] ∈ R.

pred R is antisymmetric in X **means** x ∈ X & y ∈ X & [x, y] ∈ R & [y, x] ∈ R **implies** x = y.

pred R is asymmetric in X **means** x ∈ X & y ∈ X & [x, y] ∈ R **implies not** [y, x] ∈ R.

pred R is connected in X **means** x ∈ X & y ∈ X & x ≠ y **implies** [x, y] ∈ R **or** [y, x] ∈ R.

pred R is strongly connected in X **means** x ∈ X & y ∈ X **implies** [x, y] ∈ R **or** [y, x] ∈ R.

pred R is transitive in X **means** x ∈ X & y ∈ X & z ∈ X & [x, y] ∈ R & [y, z] ∈ R **implies** [x, z] ∈ R.

Theorem RELAT_2:1. R is reflexive in X **iff for** x **st** x ∈ X **holds** [x, x] ∈ R.

Theorem RELAT_2:2. R is irreflexive in X **iff for** x **st** x ∈ X **holds not** [x, x] ∈ R.

Theorem RELAT_2:3. R is symmetric in X **iff for** x, y **st** x ∈ X & y ∈ X & [x, y] ∈ R **holds** [y, x] ∈ R.

Theorem RELAT_2:4. R is antisymmetric in X **iff for** x, y **st** x ∈ X & y ∈ X & [x, y] ∈ R & [y, x] ∈ R **holds** x = y.

Theorem RELAT_2:5. R is asymmetric in X **iff for** x, y **st** x ∈ X & y ∈ X & [x, y] ∈ R **holds not** [y, x] ∈ R.

Theorem RELAT_2:6. R is connected in X **iff for** x, y **st** x ∈ X & y ∈ X & x ≠ y **holds** [x, y] ∈ R **or** [y, x] ∈ R.

Theorem RELAT_2:7. R is strongly connected in X **iff for** x, y **st** x ∈ X & y ∈ X **holds** [x, y] ∈ R **or** [y, x] ∈ R.

Theorem RELAT_2:8. R is transitive in X **iff for** x, y, z **st** x ∈ X & y ∈ X & z ∈ X & [x, y] ∈ R & [y, z] ∈ R **holds** [x, z] ∈ R.

Definition

let R.

pred R is reflexive **means** R is reflexive in field R.

pred R is irreflexive **means** R is irreflexive in field R.

pred R is symmetric **means** R is symmetric in field R.

pred R is antisymmetric **means** R is antisymmetric in field R.

pred R is asymmetric **means** R is asymmetric in field R.

pred R is connected **means** R is connected in field R.

pred R is strongly connected **means** R is strongly connected in field R.

pred R is transitive **means** R is transitive in field R.

Theorem RELAT_2:9. R is reflexive **iff** R is reflexive in field R.

Theorem RELAT_2:10. R is irreflexive **iff** R is irreflexive in field R.

Theorem RELAT_2:11. R is symmetric **iff** R is symmetric in field R.

Theorem RELAT_2:12. R is antisymmetric **iff** R is antisymmetric in field R.

Theorem RELAT_2:13. R is asymmetric **iff** R is asymmetric in field R.

Theorem RELAT_2:14. R is connected **iff** R is connected in field R.

Theorem RELAT_2:15. R is strongly connected **iff** R is strongly connected in field R.

Theorem RELAT_2:16. R is transitive **iff** R is transitive in field R.

Theorem RELAT_2:17. R is reflexive **iff** $\triangle$field R $\subseteq$ R.

Theorem RELAT_2:18. R is irreflexive **iff** $\triangle$(field R)$\cap$R = $\emptyset$.

Theorem RELAT_2:19. R is antisymmetric in X **iff** R$\setminus\triangle$X is asymmetric in X.

Theorem RELAT_2:20. R is asymmetric in X **implies** R$\cup\triangle$X is antisymmetric in X.

Theorem RELAT_2:21. R is antisymmetric in X **implies** R$\setminus\triangle$X is asymmetric in X.

Theorem RELAT_2:22. R is symmetric & R is transitive **implies** R is reflexive.

Theorem RELAT_2:23. $\triangle$X is symmetric & $\triangle$X is transitive.

Theorem RELAT_2:24. $\triangle$X is antisymmetric & $\triangle$X is reflexive.

Theorem RELAT_2:25. R is irreflexive & R is transitive **implies** R is asymmetric.

Theorem RELAT_2:26. R is asymmetric **implies** R is irreflexive & R is antisymmetric.

Theorem RELAT_2:27. R is reflexive **implies** $R^\smile$ is reflexive.

Theorem RELAT_2:28. R is irreflexive **implies** $R^\smile$ is irreflexive.

Theorem RELAT_2:29. R is reflexive **implies** dom R = dom ($R^\smile$) & rng R = rng ($R^\smile$).

Theorem RELAT_2:30. R is symmetric **iff** R = $R^\smile$.

Theorem RELAT_2:31. P is reflexive & R is reflexive **implies** P$\cup$R is reflexive & P$\cap$R is reflexive.

Theorem RELAT_2:32. P is irreflexive & R is irreflexive **implies** P$\cup$R is irreflexive & P$\cap$R is irreflexive.

Theorem RELAT_2:33. P is irreflexive **implies** P$\setminus$R is irreflexive.

Theorem RELAT_2:34. R is symmetric **implies** $R^\smile$ is symmetric.

Theorem RELAT_2:35. P is symmetric & R is symmetric **implies** P$\cup$R is symmetric & P$\cap$R is symmetric & P$\setminus$R is symmetric.

Theorem RELAT_2:36. R is asymmetric **implies** $R^\smile$ is asymmetric.

Theorem RELAT_2:37. P is asymmetric & R is asymmetric **implies** P$\cap$R is asymmetric.

Theorem RELAT_2:38. P is asymmetric **implies** P$\setminus$R is asymmetric.

Theorem RELAT_2:39. R is antisymmetric **iff** R$\cap$($R^\smile$) $\subseteq$ $\triangle$(dom R).

Theorem RELAT_2:40. R is antisymmetric **implies** $R^{\smile}$ is antisymmetric.

Theorem RELAT_2:41. P is antisymmetric **implies** P∩R is antisymmetric & P∖R is antisymmetric.

Theorem RELAT_2:42. R is transitive **implies** $R^{\smile}$ is transitive.

Theorem RELAT_2:43. P is transitive & R is transitive **implies** P∩R is transitive.

Theorem RELAT_2:44. R is transitive **iff** R·R ⊆ R.

Theorem RELAT_2:45. R is connected **iff** ⟦field R, field R⟧∖△(field R) ⊆ R∪$R^{\smile}$.

Theorem RELAT_2:46. R is strongly connected **implies** R is connected & R is reflexive.

Theorem RELAT_2:47. R is strongly connected **iff** ⟦field R, field R⟧ = R∪$R^{\smile}$.

Chapter 15

RELSET_1

Relations Defined on Sets

by

Edmund Woronowicz [1]

Warsaw University (Białystok)

Summary. The article includes theorems concerning properties of relations defined as a subset of the Cartesian product of two sets (mode Relation of X,Y where X,Y are sets). Some notions, introduced in RELAT_1 such as domain, codomain, field of a relation, composition of relations, image and inverse image of a set under a relation are redefined.

The symbols used in this article are introduced in the following vocabularies: FAM_OP, BOOLE, REAL_1, FUNC_REL, and RELATION. The terminology and notation used in this article have been introduced in the following articles: TARSKI, BOOLE, and RELAT_1.

reserve A, B, X, X1, X2, Y, Y1, Y2, Z, W **for** set.

reserve a, b, c, d, x, y, z **for** Any.

Definition

let X, Y.

mode Relation **of** X, Y → Relation **means it** ⊆ ⟦X, Y⟧.

Theorem RELSET_1:1. **for** R **being** Relation **holds** R ⊆ ⟦X, Y⟧ **iff** R **is** Relation **of** X, Y.

[1]Supported by RPBP.III-24.C1.

reserve P, P1, P2, Q, R **for** Relation **of** X, Y.

Theorem RELSET_1:2. A ⊆ R **implies** A ⊆ ⟦X, Y⟧.

Theorem RELSET_1:3. A ⊆ ⟦X, Y⟧ **implies** A **is** Relation **of** X, Y.

Theorem RELSET_1:4. A ⊆ R **implies** A **is** Relation **of** X, Y.

Theorem RELSET_1:5. ⟦X, Y⟧ **is** Relation **of** X, Y.

Theorem RELSET_1:6. a ∈ R **implies** **ex** x, y **st** a = [x, y] & x ∈ X & y ∈ Y.

Theorem RELSET_1:7. [x, y] ∈ R **implies** x ∈ X & y ∈ Y.

Theorem RELSET_1:8. x ∈ X & y ∈ Y **implies** {[x, y]} **is** Relation **of** X, Y.

Theorem RELSET_1:9. **for** R **being** Relation **st** dom R ⊆ X **holds** R **is** Relation **of** X, rng R.

Theorem RELSET_1:10. **for** R **being** Relation **st** rng R ⊆ Y **holds** R **is** Relation **of** dom R, Y.

Theorem RELSET_1:11. **for** R **being** Relation **st** dom R ⊆ X & rng R ⊆ Y **holds** R **is** Relation **of** X, Y.

Theorem RELSET_1:12. dom R ⊆ X & rng R ⊆ Y.

Theorem RELSET_1:13. dom R ⊆ X1 **implies** R **is** Relation **of** X1, Y.

Theorem RELSET_1:14. rng R ⊆ Y1 **implies** R **is** Relation **of** X, Y1.

Theorem RELSET_1:15. X ⊆ X1 **implies** R **is** Relation **of** X1, Y.

Theorem RELSET_1:16. Y ⊆ Y1 **implies** R **is** Relation **of** X, Y1.

Theorem RELSET_1:17. X ⊆ X1 & Y ⊆ Y1 **implies** R **is** Relation **of** X1, Y1.

Definition

let X, Y, P, R.

redefine

func P∪R → Relation **of** X, Y.

func P∩R → Relation **of** X, Y.

func P∖R → Relation **of** X, Y.

Theorem RELSET_1:18. R∩⟦X, Y⟧ = R.

Definition

let X, Y, R.

redefine

func dom R → Subset **of** X.

func rng R → Subset **of** Y.

Theorem RELSET_1:19. field R ⊆ X∪Y.

Theorem RELSET_1:20. **for** R **being** Relation **holds** R **is** Relation **of** dom R, rng R.

Theorem RELSET_1:21. dom R ⊆ X1 & rng R ⊆ Y1 **implies** R **is** Relation **of** X1, Y1.

Theorem RELSET_1:22. (**for** x **st** x ∈ X **ex** y **st** [x, y] ∈ R) **iff** dom R = X.

Theorem RELSET_1:23. (**for** y **st** y ∈ Y **ex** x **st** [x, y] ∈ R) **iff** rng R = Y.

Definition

let X, Y, R.

redefine

func R⌣ → Relation **of** Y, X.

Definition

let X, Y, Z.

let P **be** Relation **of** X, Y.

let R **be** Relation **of** Y, Z.

redefine

func P·R → Relation **of** X, Z.

Theorem RELSET_1:24. dom (R⌣) = rng R & rng (R⌣) = dom R.

Theorem RELSET_1:25. ∅**is** Relation **of** X, Y.

Theorem RELSET_1:26. R **is** Relation **of** ∅, Y **implies** R = ∅.

Theorem RELSET_1:27. R **is** Relation **of** X, ∅ **implies** R = ∅.

Theorem RELSET_1:28. △X ⊆ ⟦X, X⟧.

Theorem RELSET_1:29. △X **is** Relation **of** X, X.

Theorem RELSET_1:30. △A ⊆ R **implies** A ⊆ dom R & A ⊆ rng R.

Theorem RELSET_1:31. △X ⊆ R **implies** X = dom R & X ⊆ rng R.

Theorem RELSET_1:32. △Y ⊆ R **implies** Y ⊆ dom R & Y = rng R.

Definition

let X, Y, R, A.

redefine

func R↾A → Relation **of** X, Y.

Definition

let X, Y, B, R.

redefine

func B↾R → Relation **of** X, Y.

Theorem RELSET_1:33. R↾X1 **is** Relation **of** X1, Y.

Theorem RELSET_1:34. X ⊆ X1 **implies** R↾X1 = R.

Theorem RELSET_1:35. Y1↾R **is** Relation **of** X, Y1.

Theorem RELSET_1:36. Y ⊆ Y1 **implies** Y1↾R = R.

Definition

let X, Y, R, A.

redefine

func R.A → Subset **of** Y.

func $R^{-1}A \to$ Subset **of** X.

Theorem RELSET_1:37. R.A ⊆ Y & $R^{-1}A$ ⊆ X.

Theorem RELSET_1:38. R.X = rng R & $R^{-1}Y$ = dom R.

Theorem RELSET_1:39. R.($R^{-1}Y$) = rng R & R^{-1}(R.X) = dom R.

scheme Rel_On_Set_Ex{A() → set, B() → set, P[Any, Any]}: **ex** R **being** Relation **of** A(), B() **st** **for** x, y **holds** [x, y] ∈ R **iff** x ∈ A() & y ∈ B() & P[x, y].

Definition

let X.

mode Relation **of** X → Relation **of** X, X **means** **it** ⊆ ⟦X, X⟧.

Theorem RELSET_1:40. **for** R **being** Relation **of** X, X **holds** R ⊆ ⟦X, X⟧ **iff** R **is** Relation **of** X.

reserve P, Q, R **for** Relation **of** X.

Theorem RELSET_1:41. ⟦X, X⟧ **is** Relation **of** X.

Theorem RELSET_1:42. **for** R **being** Relation **of** X, X **st** dom R = X & rng R = X **holds** R **is** Relation **of** X.

Theorem RELSET_1:43. △X **is** Relation **of** X.

Theorem RELSET_1:44. △X ⊆ R **implies** X = dom R & X = rng R.

Theorem RELSET_1:45. R·(△X) = R & (△X)·R = R.

reserve D, D1, D2, E, E1, F **for** DOMAIN.

reserve P, P1, Q, R **for** Relation **of** D, E.

reserve a, x, x1 **for** Element **of** D.

reserve b, y, y1 **for** Element **of** E.

reserve c, z **for** Element **of** F.

Theorem RELSET_1:46. △D ≠ ∅.

Definition

let D, E, R.

redefine

func dom R → Element **of** bool D.

func rng R → Element **of** bool E.

Theorem RELSET_1:47. **for** x **being** Element **of** D **holds** x ∈ dom R **iff** **ex** y **being** Element **of** E **st** [x, y] ∈ R.

Theorem RELSET_1:48. **for** y **being** Element **of** E **holds** y ∈ rng R **iff** **ex** x **being** Element **of** D **st** [x, y] ∈ R.

Theorem RELSET_1:49. **for** x **being** Element **of** D **holds** x ∈ dom R **implies** **ex** y **being** Element **of** E **st** y ∈ rng R.

Theorem RELSET_1:50. **for** y **being** Element **of** E **holds** y ∈ rng R **implies** **ex** x **being** Element **of** D **st** x ∈ dom R.

Theorem RELSET_1:51. **for** P **being** (Relation **of** D, E), R **being** (Relation **of** E, F) **for** x **being** (Element **of** D), z **being** Element **of** F **holds** [x, z] ∈ P·R **iff** **ex** y **being** Element **of** E **st** [x, y] ∈ P & [y, z] ∈ R.

Definition

let D, E, R, D1.

redefine

func R**.**D1 → Element **of** bool E.

func R^{-1}D1 → Element **of** bool D.

Theorem RELSET_1:52. y ∈ R**.**D1 **iff** **ex** x **being** Element **of** D **st** [x, y] ∈ R & x ∈ D1.

Theorem RELSET_1:53. x ∈ R^{-1}D2 **iff** **ex** y **being** Element **of** E **st** [x, y] ∈ R & y ∈ D2.

scheme Rel_On_Dom_Ex{A() → DOMAIN, B() → DOMAIN, P[Any, Any]}: **ex** R **being** Relation **of** A(), B() **st** **for** x **being** (Element **of** A()), y **being** Element **of** B() **holds** [x, y] ∈ R **iff** x ∈ A() & y ∈ B() & P[x, y].

Chapter 16

WELLORD1

The Well Ordering Relations

by

Grzegorz Bancerek [1]

Warsaw University (Białystok)

Summary. Some theorems about well ordering relations are proved. The goal of the article is to prove that any two well ordering relations are either isomorphic or one of them is isomorphic to a segment of the other. The following concepts are defined: the segment of a relation induced by an element, well founded relations, well ordering relations, the restriction of a relation to a set, and the isomorphism of two relations. A number of simple facts is presented.

The symbols used in this article are introduced in the following vocabularies: BOOLE, FAM_OP, REAL_1, FUNC_REL, RELATION, REL_REL, WELLORD, and FUNC. The terminology and notation used in this article have been introduced in the following articles: TARSKI, BOOLE, ENUMSET1, RELAT_1, RELAT_2, and FUNCT_1.

reserve a, b, c, d, e, x, y, z **for** Any, X, Y, Z **for** set.

scheme Extensionality{A() → set, B() → set, P[Any]}: A() = B() **provided** A: **for** a **holds** a ∈ A() **iff** P[a] **and** B: **for** a **holds** a ∈ B() **iff** P[a].

reserve R, S, T **for** Relation.

Definition

let R, a.

[1]Supported by RPBP.III-24.C1.

func R-Seg(a) → set **means** x ∈ **it** **iff** x ≠ a & [x, a] ∈ R.

Theorem WELLORD1:1. **for** R, Y, a **holds** Y = R-Seg(a) **iff** **for** b **holds** b ∈ Y **iff** b ≠ a & [b, a] ∈ R.

Theorem WELLORD1:2. x ∈ field R **or** R-Seg(x) = ∅.

Definition

let R.

pred R is well founded **means** **for** Y **st** Y ⊆ field R & Y ≠ ∅ **ex** a **st** a ∈ Y & R-Seg(a)∩Y = ∅.

let X.

pred R is well founded in X **means** **for** Y **st** Y ⊆ X & Y ≠ ∅ **ex** a **st** a ∈ Y & R-Seg(a)∩Y = ∅.

Theorem WELLORD1:3. **for** R **holds** R is well founded **iff** **for** Y **st** Y ⊆ field R & Y ≠ ∅ **ex** a **st** a ∈ Y & R-Seg(a)∩Y = ∅.

Theorem WELLORD1:4. **for** R, X **holds** R is well founded in X **iff** **for** Y **st** Y ⊆ X & Y ≠ ∅ **ex** a **st** a ∈ Y & R-Seg(a)∩Y = ∅.

Theorem WELLORD1:5. R is well founded **iff** R is well founded in field R.

Definition

let R.

pred R is well-ordering-relation **means** R is reflexive & R is transitive & R is antisymmetric & R is connected & R is well founded.

let X.

pred R well orders X **means** R is reflexive in X & R is transitive in X & R is antisymmetric in X & R is connected in X & R is well founded in X.

Theorem WELLORD1:6. **for** R **holds** R is well-ordering-relation **iff** R is reflexive & R is transitive & R is antisymmetric & R is connected & R is well founded.

Theorem WELLORD1:7. **for** R, X **holds** R well orders X **iff** R is reflexive in X & R is transitive in X & R is antisymmetric in X & R is connected in X & R is well founded in X.

Theorem WELLORD1:8. R well orders field R **iff** R is well-ordering-relation.

Theorem WELLORD1:9. R well orders X **implies** **for** Y **st** Y ⊆ X & Y ≠ ∅ **ex** a **st** a ∈ Y & **for** b **st** b ∈ Y **holds** [a, b] ∈ R.

Theorem WELLORD1:10. R is well-ordering-relation **implies** **for** Y **st** Y ⊆ field R & Y ≠ ∅ **ex** a **st** a ∈ Y & **for** b **st** b ∈ Y **holds** [a, b] ∈ R.

Theorem WELLORD1:11. **for** R **st** R is well-ordering-relation & field R ≠ ∅ **ex** a **st** a ∈ field R & **for** b **st** b ∈ field R **holds** [a, b] ∈ R.

Theorem WELLORD1:12. **for** R **st** R is well-ordering-relation & field R ≠ ∅ **for** a **st** a ∈ field R **holds** (**for** b **st** b ∈ field R **holds** [b, a] ∈ R) **or** (**ex** b **st** b ∈ field R & [a, b] ∈ R & **for** c **st** c ∈ field R & [a, c] ∈ R **holds** c = a **or** [b, c] ∈ R).

reserve F, G, H **for** Function.

Theorem WELLORD1:13. R-Seg(a) ⊆ field R.

Definition

let R, Y.

func $R|^2Y$ → Relation **means it** = R∩⟦Y, Y⟧.

Theorem WELLORD1:14. $R|^2Y$ = R∩⟦Y, Y⟧.

Theorem WELLORD1:15. $R|^2X$ ⊆ R & $R|^2X$ ⊆ ⟦X, X⟧.

Theorem WELLORD1:16. x ∈ $R|^2X$ **iff** x ∈ R & x ∈ ⟦X, X⟧.

Theorem WELLORD1:17. $R|^2X$ = X|R|X.

Theorem WELLORD1:18. $R|^2X$ = X|(R|X).

Theorem WELLORD1:19. x ∈ field ($R|^2X$) **implies** x ∈ field R & x ∈ X.

Theorem WELLORD1:20. field ($R|^2X$) ⊆ field R & field ($R|^2X$) ⊆ X.

Theorem WELLORD1:21. ($R|^2X$)-Seg(a) ⊆ R-Seg(a).

Theorem WELLORD1:22. R is reflexive **implies** $R|^2X$ is reflexive.

Theorem WELLORD1:23. R is connected **implies** $R|^2Y$ is connected.

Theorem WELLORD1:24. R is transitive **implies** $R|^2Y$ is transitive.

Theorem WELLORD1:25. R is antisymmetric **implies** $R|^2Y$ is antisymmetric.

Theorem WELLORD1:26. ($R|^2X$)$|^2Y$ = $R|^2$(X∩Y).

Theorem WELLORD1:27. ($R|^2X$)$|^2Y$ = ($R|^2Y$)$|^2X$.

Theorem WELLORD1:28. ($R|^2Y$)$|^2Y$ = $R|^2Y$.

Theorem WELLORD1:29. Z ⊆ Y **implies** ($R|^2Y$)$|^2Z$ = $R|^2Z$.

Theorem WELLORD1:30. $R|^2$field R = R.

Theorem WELLORD1:31. R is well founded **implies** $R|^2X$ is well founded.

Theorem WELLORD1:32. R is well-ordering-relation **implies** $R|^2Y$ is well-ordering-relation.

Theorem WELLORD1:33. R is well-ordering-relation **implies** R-Seg(a) ⊆ R-Seg(b) **or** R-Seg(b) ⊆ R-Seg(a).

Theorem WELLORD1:34. R is well-ordering-relation **implies** $R|^2$(R-Seg(a)) is well-ordering-relation.

Theorem WELLORD1:35. R is well-ordering-relation & a ∈ field R & b ∈ R-Seg(a) **implies** ($R|^2$(R-Seg(a)))-Seg(b) = R-Seg(b).

Theorem WELLORD1:36. R is well-ordering-relation & Y ⊆ field R **implies** (Y = field R **or** (**ex** a **st** a ∈ field R & Y = R-Seg(a)) **iff for** a **st** a ∈ Y **for** b **st** [b, a] ∈ R **holds** b ∈ Y).

Theorem WELLORD1:37. R is well-ordering-relation & a ∈ field R & b ∈ field R **implies** ([a, b] ∈ R **iff** R-Seg(a) ⊆ R-Seg(b)).

Theorem WELLORD1:38. R is well-ordering-relation & a ∈ field R & b ∈ field R **implies** (R-Seg(a) ⊆ R-Seg(b) **iff** a = b **or** a ∈ R-Seg(b)).

Theorem WELLORD1:39. R is well-ordering-relation & X ⊆ field R **implies** field ($R|^2$X) = X.

Theorem WELLORD1:40. R is well-ordering-relation **implies** field ($R|^2$R-Seg(a)) = R-Seg(a).

Theorem WELLORD1:41. R is well-ordering-relation **implies for** Z **st for** a **st** a ∈ field R & R-Seg(a) ⊆ Z **holds** a ∈ Z **holds** field R ⊆ Z.

Theorem WELLORD1:42. R is well-ordering-relation & a ∈ field R & b ∈ field R & (**for** c **st** c ∈ R-Seg(a) **holds** [c, b] ∈ R & c ≠ b) **implies** [a, b] ∈ R.

Theorem WELLORD1:43. R is well-ordering-relation & dom F = field R & rng F ⊆ field R & (**for** a, b **st** [a, b] ∈ R & a ≠ b **holds** [F.a, F.b] ∈ R & F.a ≠ F.b) **implies for** a **st** a ∈ field R **holds** [a, F.a] ∈ R.

Definition

let R, S, F.

pred F is isomorphism of R, S **means** dom F = field R & rng F = field S & F is 1-1 & **for** a, b **holds** [a, b] ∈ R **iff** a ∈ field R & b ∈ field R & [F.a, F.b] ∈ S.

Theorem WELLORD1:44. F is isomorphism of R, S **iff** dom F = field R & rng F = field S & F is 1-1 & **for** a, b **holds** [a, b] ∈ R **iff** a ∈ field R & b ∈ field R & [F.a, F.b] ∈ S.

Theorem WELLORD1:45. F is isomorphism of R, S **implies for** a, b **st** [a, b] ∈ R & a ≠ b **holds** [F.a, F.b] ∈ S & F.a ≠ F.b.

Definition

let R, S.

pred R, S are isomorphic **means ex** F **st** F is isomorphism of R, S.

Theorem WELLORD1:46. R, S are isomorphic **iff ex** F **st** F is isomorphism of R, S.

Theorem WELLORD1:47. Id (field R) is isomorphism of R, R.

Theorem WELLORD1:48. R, R are isomorphic.

Theorem WELLORD1:49. F is isomorphism of R, S **implies** F^{-1} is isomorphism of S, R.

Theorem WELLORD1:50. R, S are isomorphic **implies** S, R are isomorphic.

Theorem WELLORD1:51. F is isomorphism of R, S & G is isomorphism of S, T **implies** G·F is isomorphism of R, T.

Theorem WELLORD1:52. R, S are isomorphic & S, T are isomorphic **implies** R, T are isomorphic.

Theorem WELLORD1:53. F is isomorphism of R, S **implies** (R is reflexive **implies** S is reflexive) & (R is transitive **implies** S is transitive) & (R is connected **implies** S is connected) & (R is antisymmetric **implies** S is antisymmetric) & (R is well founded **implies** S is well founded).

Theorem WELLORD1:54. R is well-ordering-relation & F is isomorphism of R, S **implies** S is well-ordering-relation.

Theorem WELLORD1:55. R is well-ordering-relation **implies for** F, G **st** F is isomorphism of R, S & G is isomorphism of R, S **holds** F = G.

Definition

let R, S.

assume R is well-ordering-relation & R, S are isomorphic.

func canonical isomorphism of (R, S) → Function **means it** is isomorphism of R, S.

Theorem WELLORD1:56. R is well-ordering-relation & R, S are isomorphic **implies** (F = canonical isomorphism of (R, S) **iff** F is isomorphism of R, S).

Theorem WELLORD1:57. R is well-ordering-relation **implies for** a **st** a ∈ field R **holds not** R, R$\restriction^2$(R-Seg(a)) are isomorphic.

Theorem WELLORD1:58. R is well-ordering-relation & a ∈ field R & b ∈ field R & a ≠ b **implies not** R$\restriction^2$(R-Seg(a)), R$\restriction^2$(R-Seg(b)) are isomorphic.

Theorem WELLORD1:59. R is well-ordering-relation & Z ⊆ field R & F is isomorphism of R, S **implies** F$\restriction$Z is isomorphism of R$\restriction^2$Z, S$\restriction^2$(F.Z) & R$\restriction^2$Z, S$\restriction^2$(F.Z) are isomorphic.

Theorem WELLORD1:60. R is well-ordering-relation & F is isomorphism of R, S **implies for** a **st** a ∈ field R **ex** b **st** b ∈ field S & F.(R-Seg(a)) = S-Seg(b).

Theorem WELLORD1:61. R is well-ordering-relation & F is isomorphism of R, S **implies for** a **st** a ∈ field R **ex** b **st** b ∈ field S & R$\restriction^2$(R-Seg(a)), S$\restriction^2$(S-Seg(b)) are isomorphic.

Theorem WELLORD1:62. R is well-ordering-relation & S is well-ordering-relation & a ∈ field R & b ∈ field S & c ∈ field S & R, S$\restriction^2$(S-Seg(b)) are isomorphic & R$\restriction^2$(R-Seg(a)), S$\restriction^2$ (S-Seg(c)) are isomorphic **implies** S-Seg(c) ⊆ S-Seg(b) & [c, b] ∈ S.

Theorem WELLORD1:63. R is well-ordering-relation & S is well-ordering-relation **implies** R, S are isomorphic **or** (**ex** a **st** a ∈ field R & R$\restriction^2$(R-Seg(a)), S are isomorphic) **or** (**ex** a **st** a ∈ field S & R, S$\restriction^2$(S-Seg(a)) are isomorphic).

Theorem WELLORD1:64. Y ⊆ field R & R is well-ordering-relation **implies** R, R$\restriction^2$Y are isomorphic **or ex** a **st** a ∈ field R & R$\restriction^2$(R-Seg(a)), R$\restriction^2$Y are isomorphic.

Chapter 17

SETFAM_1

Families of Sets

by

Beata Padlewska [1]

Warsaw University (Białystok)

Summary. The article contains definitions of the following concepts: family of sets, family of subsets of a set, the intersection of a family of sets. Functions $\cap$, $\cup$, and $\setminus$ are redefined for families of subsets of a set. Some properties of these notions are presented.

The symbols used in this article are introduced in the following vocabularies: BOOLE, FAM_OP, SUB_OP, and SFAMILY. The terminology and notation used in this article have been introduced in the following articles: TARSKI, BOOLE, ENUMSET1, and SUBSET_1.

reserve X, X1, X2, X3, Y, Z, Z1, Z2, D **for** set, x, y, z **for** Any.

Definition

let X.

func $\bigcap$X $\rightarrow$ set **means for** x **holds** x $\in$ **it iff** (**for** Y **holds** Y $\in$ X **implies** x $\in$ Y) **if** X $\neq \emptyset$ **otherwise it** $= \emptyset$.

Theorem SETFAM_1:1. X $\neq \emptyset$ **implies for** x **holds** x $\in \bigcap$X **iff for** Y **st** Y $\in$ X **holds** x $\in$ Y.

Theorem SETFAM_1:2. $\bigcap\emptyset = \emptyset$.

[1] Supported by RPBP.III-24.C1.

Theorem SETFAM_1:3. $\bigcap$X $\subseteq$ $\bigcup$X.

Theorem SETFAM_1:4. Z $\in$ X **implies** $\bigcap$X $\subseteq$ Z.

Theorem SETFAM_1:5. $\emptyset$ $\in$ X **implies** $\bigcap$X = $\emptyset$.

Theorem SETFAM_1:6. X $\neq$ $\emptyset$ & (**for** Z1 **st** Z1 $\in$ X **holds** Z $\subseteq$ Z1) **implies** Z $\subseteq$ $\bigcap$X.

Theorem SETFAM_1:7. X $\neq$ $\emptyset$ & X $\subseteq$ Y **implies** $\bigcap$Y $\subseteq$ $\bigcap$X.

Theorem SETFAM_1:8. X $\in$ Y & X $\subseteq$ Z **implies** $\bigcap$Y $\subseteq$ Z.

Theorem SETFAM_1:9. X $\in$ Y & X$\cap$Z = $\emptyset$ **implies** $\bigcap$Y$\cap$Z = $\emptyset$.

Theorem SETFAM_1:10. X $\neq$ $\emptyset$ & Y $\neq$ $\emptyset$ **implies** $\bigcap$(X$\cup$Y) = $\bigcap$X$\cap\bigcap$Y.

Theorem SETFAM_1:11. $\bigcap$\{x\} = x.

Theorem SETFAM_1:12. $\bigcap$\{X, Y\} = X$\cap$Y.

Definition

mode Set-Family $\rightarrow$ set **means not contradiction**.

reserve SFX, SFY, SFZ **for** Set-Family.

Theorem SETFAM_1:13. x **is** Set-Family.

Theorem SETFAM_1:14. SFX = SFY **iff** (**for** X **holds** X $\in$ SFX **iff** X $\in$ SFY).

Definition

let SFX, SFY.

pred SFX is finer than SFY **means for** X **st** X $\in$ SFX **ex** Y **st** Y $\in$ SFY & X $\subseteq$ Y.

pred SFX is coarser than SFY **means for** Y **st** Y $\in$ SFY **ex** X **st** X $\in$ SFX & X $\subseteq$ Y.

Theorem SETFAM_1:15. SFX is finer than SFY **iff for** X **st** X $\in$ SFX **ex** Y **st** Y $\in$ SFY & X $\subseteq$ Y.

Theorem SETFAM_1:16. SFX is coarser than SFY **iff for** Y **st** Y $\in$ SFY **ex** X **st** X $\in$ SFX & X $\subseteq$ Y.

Theorem SETFAM_1:17. SFX $\subseteq$ SFY **implies** SFX is finer than SFY.

Theorem SETFAM_1:18. SFX is finer than SFY **implies** $\bigcup$SFX $\subseteq$ $\bigcup$SFY.

Theorem SETFAM_1:19. SFY $\neq$ $\emptyset$ & SFX is coarser than SFY **implies** $\bigcap$SFX $\subseteq$ $\bigcap$SFY.

Definition

redefine

func $\emptyset$ $\rightarrow$ Set-Family.

let x.

func \{x\} $\rightarrow$ Set-Family.

let y.

func \{x, y\} $\rightarrow$ Set-Family.

Theorem SETFAM_1:20. ∅ is finer than SFX.

Theorem SETFAM_1:21. SFX is finer than ∅ **implies** SFX = ∅.

Theorem SETFAM_1:22. SFX is finer than SFX.

Theorem SETFAM_1:23. SFX is finer than SFY & SFY is finer than SFZ **implies** SFX is finer than SFZ.

Theorem SETFAM_1:24. SFX is finer than {Y} **implies for** X **st** X ∈ SFX **holds** X ⊆ Y.

Theorem SETFAM_1:25. SFX is finer than {X, Y} **implies for** Z **st** Z ∈ SFX **holds** Z ⊆ X **or** Z ⊆ Y.

Definition

let SFX, SFY.

func ⋓(SFX, SFY) → Set-Family **means** Z ∈ **it iff ex** X, Y **st** X ∈ SFX & Y ∈ SFY & Z = X∪Y.

func ⋒(SFX, SFY) → Set-Family **means** Z ∈ **it iff ex** X, Y **st** X ∈ SFX & Y ∈ SFY & Z = X∩Y.

func ∖∖(SFX, SFY) → Set-Family **means** Z ∈ **it iff ex** X, Y **st** X ∈ SFX & Y ∈ SFY & Z = X∖Y.

Theorem SETFAM_1:26. Z ∈ ⋓(SFX, SFY) **iff ex** X, Y **st** X ∈ SFX & Y ∈ SFY & Z = X∪Y.

Theorem SETFAM_1:27. Z ∈ ⋒(SFX, SFY) **iff ex** X, Y **st** X ∈ SFX & Y ∈ SFY & Z = X∩Y.

Theorem SETFAM_1:28. Z ∈ ∖∖(SFX, SFY) **iff ex** X, Y **st** X ∈ SFX & Y ∈ SFY & Z = X∖Y.

Theorem SETFAM_1:29. SFX is finer than ⋓(SFX, SFX).

Theorem SETFAM_1:30. ⋒(SFX, SFX) is finer than SFX.

Theorem SETFAM_1:31. ∖∖(SFX, SFX) is finer than SFX.

Theorem SETFAM_1:32. ⋓(SFX, SFY) = ⋓(SFY, SFX).

Theorem SETFAM_1:33. ⋒(SFX, SFY) = ⋒(SFY, SFX).

Theorem SETFAM_1:34. SFX∩SFY ≠ ∅ **implies** ⋂SFX∩⋂SFY = ⋂⋒(SFX, SFY).

Theorem SETFAM_1:35. SFY ≠ ∅ **implies** X∪⋂SFY = ⋂⋓({X}, SFY).

Theorem SETFAM_1:36. X∩⋃SFY = ⋃⋒({X}, SFY).

Theorem SETFAM_1:37. SFY ≠ ∅ **implies** X∖⋃SFY = ⋂∖∖({X}, SFY).

Theorem SETFAM_1:38. SFY ≠ ∅ **implies** X∖⋂SFY = ⋃∖∖({X}, SFY).

Theorem SETFAM_1:39. ⋃⋒(SFX, SFY) ⊆ ⋃SFX∩⋃SFY.

Theorem SETFAM_1:40. SFX ≠ ∅ & SFY ≠ ∅ **implies** ⋂SFX∪⋂SFY ⊆ ⋂⋓(SFX, SFY).

Theorem SETFAM_1:41. SFX $\neq \emptyset$ & SFY $\neq \emptyset$ **implies** $\bigcap\setminus\setminus$(SFX, SFY) $\subseteq \bigcap$SFX$\setminus$ $\bigcap$SFY.

Definition

let D **be** set.

mode Subset-Family **of** D $\rightarrow$ Subset **of** bool D **means not contradiction**.

Theorem SETFAM_1:42. **for** F **being** Subset **of** bool D **holds** F **is** Subset-Family **of** D.

reserve F, G **for** Subset-Family **of** D.

reserve P, Q **for** Subset **of** D.

Definition

let D, F, G.

redefine

func F$\cup$G $\rightarrow$ Subset-Family **of** D.

func F$\cap$G $\rightarrow$ Subset-Family **of** D.

func F$\setminus$G $\rightarrow$ Subset-Family **of** D.

Theorem SETFAM_1:43. X $\in$ F **implies** X **is** Subset **of** D.

Definition

let D, F.

redefine

func $\bigcup$F $\rightarrow$ Subset **of** D.

Definition

let D, F.

redefine

func $\bigcap$F $\rightarrow$ Subset **of** D.

Theorem SETFAM_1:44. F = G **iff** (**for** P **holds** P $\in$ F **iff** P $\in$ G).

scheme SubFamEx{A() $\rightarrow$ set, P[Subset **of** A()]}: **ex** F **being** Subset-Family **of** A() **st for** B **being** Subset **of** A() **holds** B $\in$ F **iff** P[B].

Definition

let D, F.

func F^c $\rightarrow$ Subset-Family **of** D **means for** P **being** Subset **of** D **holds** P $\in$ **it iff** $P^c \in$ F.

Theorem SETFAM_1:45. **for** P **holds** P $\in F^c$ **iff** $P^c \in$ F.

Theorem SETFAM_1:46. F $\neq \emptyset$ **implies** $F^c \neq \emptyset$.

Theorem SETFAM_1:47. F $\neq \emptyset$ **implies** ΩD$\setminus\bigcup$F = $\bigcap(F^c)$.

Theorem SETFAM_1:48. F $\neq \emptyset$ **implies** $\bigcup F^c = \Omega$D$\setminus\bigcap$F.

Chapter 18

MCART_1

Tuples, Projections and Cartesian Products

by

Andrzej Trybulec [1]

Warsaw University (Białystok)

Summary. The purpose of this article is to define projections of ordered pairs, and to introduce triples and quadruples, and their projections. The theorems in this paper may be roughly divided into two groups: theorems describing basic properties of introduced concepts and theorems related to the regularity, analogous to those proved for ordered pairs in *Some Basic Properties of Sets* by Cz. Byliński (ZFMISC_1). Cartesian products of subsets are redefined as subsets of Cartesian products.

The symbols used in this article are introduced in the following vocabularies: FAM_OP, BOOLE, and COORD. The terminology and notation used in this article have been introduced in the following articles: TARSKI, BOOLE, ENUMSET1, SUBSET_1, FUNCT_1, and ORDINAL1.

reserve v, x, x1, x2, x3, x4, y, y1, y2, y3, y4, z, z1, z2 **for** Any, X, X1, X2, X3, X4, X5, X6, Y, Y1, Y2, Y3, Y4, Y5, Z, Z1, Z2, Z3, Z4, Z5 **for** set.

Theorem MCART_1:1. X $\neq \emptyset$ **implies ex** Y **st** Y $\in$ X & Y misses X.

Theorem MCART_1:2. X $\neq \emptyset$ **implies ex** Y **st** Y $\in$ X & **for** Y1 **st** Y1 $\in$ Y **holds** Y1 misses X.

[1]Supported by RPBP.III-24.C1.

Theorem MCART_1:3. X ≠ ∅ **implies** **ex** Y **st** Y ∈ X & **for** Y1, Y2 **st** Y1 ∈ Y2 & Y2 ∈ Y **holds** Y1 misses X.

Theorem MCART_1:4. X ≠ ∅ **implies** **ex** Y **st** Y ∈ X & **for** Y1, Y2, Y3 **st** Y1 ∈ Y2 & Y2 ∈ Y3 & Y3 ∈ Y **holds** Y1 misses X.

Theorem MCART_1:5. X ≠ ∅ **implies** **ex** Y **st** Y ∈ X & **for** Y1, Y2, Y3, Y4 **st** Y1 ∈ Y2 & Y2 ∈ Y3 & Y3 ∈ Y4 & Y4 ∈ Y **holds** Y1 misses X.

Theorem MCART_1:6. X ≠ ∅ **implies** **ex** Y **st** Y ∈ X & **for** Y1, Y2, Y3, Y4, Y5 **st** Y1 ∈ Y2 & Y2 ∈ Y3 & Y3 ∈ Y4 & Y4 ∈ Y5 & Y5 ∈ Y **holds** Y1 misses X.

Definition

let x.

given x1, x2 **being** Any **such that** x = [x1, x2].

func x_1 **means** x = [y1, y2] **implies** **it** = y1.

func x_2 **means** x = [y1, y2] **implies** **it** = y2.

Theorem MCART_1:7. $[x, y]_1$ = x & $[x, y]_2$ = y.

Theorem MCART_1:8. (**ex** x, y **st** z = [x, y]) **implies** [z_1, z_2] = z.

Theorem MCART_1:9. X ≠ ∅ **implies** **ex** v **st** v ∈ X & **not** **ex** x, y **st** (x ∈ X **or** y ∈ X) & v = [x, y].

Theorem MCART_1:10. z ∈ ⟦X, Y⟧ **implies** z_1 ∈ X & z_2 ∈ Y.

Theorem MCART_1:11. (**ex** x, y **st** z = [x, y]) & z_1 ∈ X & z_2 ∈ Y **implies** z ∈ ⟦X, Y⟧.

Theorem MCART_1:12. z ∈ ⟦{x}, Y⟧ **implies** z_1 = x & z_2 ∈ Y.

Theorem MCART_1:13. z ∈ ⟦X, {y}⟧ **implies** z_1 ∈ X & z_2 = y.

Theorem MCART_1:14. z ∈ ⟦{x}, {y}⟧ **implies** z_1 = x & z_2 = y.

Theorem MCART_1:15. z ∈ ⟦{x1, x2}, Y⟧ **implies** (z_1 = x1 **or** z_1 = x2) & z_2 ∈ Y.

Theorem MCART_1:16. z ∈ ⟦X, {y1, y2}⟧ **implies** z_1 ∈ X & (z_2 = y1 **or** z_2 = y2).

Theorem MCART_1:17. z ∈ ⟦{x1, x2}, {y}⟧ **implies** (z_1 = x1 **or** z_1 = x2) & z_2 = y.

Theorem MCART_1:18. z ∈ ⟦{x}, {y1, y2}⟧ **implies** z_1 = x & (z_2 = y1 **or** z_2 = y2).

Theorem MCART_1:19. z ∈ ⟦{x1, x2}, {y1, y2}⟧ **implies** (z_1 = x1 **or** z_1 = x2) & (z_2 = y1 **or** z_2 = y2).

Theorem MCART_1:20. (**ex** y, z **st** x = [y, z]) **implies** x ≠ x_1 & x ≠ x_2.

reserve xx, xx1, xx2 **for** Element **of** X.

reserve yy, yy1, yy2 **for** Element **of** Y.

Theorem MCART_1:21. X ≠ ∅ & Y ≠ ∅ **implies** [xx, yy] ∈ ⟦X, Y⟧.

Theorem MCART_1:22. X ≠ ∅ & Y ≠ ∅ **implies** [xx, yy] **is** Element **of** ⟦X, Y⟧.

Theorem MCART_1:23. x ∈ ⟦X, Y⟧ **implies** x = [x_1, x_2].

Theorem MCART_1:24. $X \neq \emptyset$ & $Y \neq \emptyset$ **implies for** x **being** Element **of** $[\![X, Y]\!]$ **holds** $x = [x_1, x_2]$.

Theorem MCART_1:25. $[\![\{x1, x2\}, \{y1, y2\}]\!] = \{[x1, y1], [x1, y2], [x2, y1], [x2, y2]\}$.

Theorem MCART_1:26. $X \neq \emptyset$ & $Y \neq \emptyset$ **implies for** x **being** Element **of** $[\![X, Y]\!]$ **holds** $x \neq x_1$ & $x \neq x_2$.

Definition

let x1, x2, x3.

func [x1, x2, x3] **means it** = [[x1, x2], x3].

Theorem MCART_1:27. [x1, x2, x3] = [[x1, x2], x3].

Theorem MCART_1:28. [x1, x2, x3] = [y1, y2, y3] **implies** x1 = y1 & x2 = y2 & x3 = y3.

Theorem MCART_1:29. $X \neq \emptyset$ **implies ex** v **st** $v \in X$ & **not ex** x, y, z **st** ($x \in X$ **or** $y \in X$) & v = [x, y, z].

Definition

let x1, x2, x3, x4.

func [x1, x2, x3, x4] **means it** = [[x1, x2, x3], x4].

Theorem MCART_1:30. [x1, x2, x3, x4] = [[x1, x2, x3], x4].

Theorem MCART_1:31. [x1, x2, x3, x4] = [[[x1, x2], x3], x4].

Theorem MCART_1:32. [x1, x2, x3, x4] = [[x1, x2], x3, x4].

Theorem MCART_1:33. [x1, x2, x3, x4] = [y1, y2, y3, y4] **implies** x1 = y1 & x2 = y2 & x3 = y3 & x4 = y4.

Theorem MCART_1:34. $X \neq \emptyset$ **implies ex** v **st** $v \in X$ & **not ex** x1, x2, x3, x4 **st** ($x1 \in X$ **or** $x2 \in X$) & v = [x1, x2, x3, x4].

Theorem MCART_1:35. $X1 \neq \emptyset$ & $X2 \neq \emptyset$ & $X3 \neq \emptyset$ **iff** $[\![X1, X2, X3]\!] \neq \emptyset$.

reserve xx1 **for** (Element **of** X1), xx2 **for** (Element **of** X2), xx3 **for** (Element **of** X3).

Theorem MCART_1:36. $X1 \neq \emptyset$ & $X2 \neq \emptyset$ & $X3 \neq \emptyset$ **implies** ($[\![X1, X2, X3]\!] = [\![Y1, Y2, Y3]\!]$ **implies** X1 = Y1 & X2 = Y2 & X3 = Y3).

Theorem MCART_1:37. $[\![X1, X2, X3]\!] \neq \emptyset$ & $[\![X1, X2, X3]\!] = [\![Y1, Y2, Y3]\!]$ **implies** X1 = Y1 & X2 = Y2 & X3 = Y3.

Theorem MCART_1:38. $[\![X, X, X]\!] = [\![Y, Y, Y]\!]$ **implies** X = Y.

Theorem MCART_1:39. $[\![\{x1\}, \{x2\}, \{x3\}]\!] = \{[x1, x2, x3]\}$.

Theorem MCART_1:40. $[\![\{x1, y1\}, \{x2\}, \{x3\}]\!] = \{[x1, x2, x3], [y1, x2, x3]\}$.

Theorem MCART_1:41. $[\![\{x1\}, \{x2, y2\}, \{x3\}]\!] = \{[x1, x2, x3], [x1, y2, x3]\}$.

Theorem MCART_1:42. $[\![\{x1\}, \{x2\}, \{x3, y3\}]\!] = \{[x1, x2, x3], [x1, x2, y3]\}$.

Theorem MCART_1:43. $[\![\{x1, y1\}, \{x2, y2\}, \{x3\}]\!] = \{[x1, x2, x3], [y1, x2, x3], [x1, y2, x3], [y1, y2, x3]\}$.

Theorem MCART_1:44. ⟦{x1, y1}, {x2}, {x3, y3}⟧ = {[x1, x2, x3], [y1, x2, x3], [x1, x2, y3], [y1, x2, y3]}.

Theorem MCART_1:45. ⟦{x1}, {x2, y2}, {x3, y3}⟧ = {[x1, x2, x3], [x1, y2, x3], [x1, x2, y3], [x1, y2, y3]}.

Theorem MCART_1:46. ⟦{x1, y1}, {x2, y2}, {x3, y3}⟧ = {[x1, x2, x3], [x1, y2, x3], [x1, x2, y3], [x1, y2, y3], [y1, x2, x3], [y1, y2, x3], [y1, x2, y3], [y1, y2, y3]}.

Definition

let X1, X2, X3.

assume X1 ≠ ∅ & X2 ≠ ∅ & X3 ≠ ∅.

let x **be** Element **of** ⟦X1, X2, X3⟧.

func x_1 → Element **of** X1 **means** x = [x1, x2, x3] **implies it** = x1.

func x_2 → Element **of** X2 **means** x = [x1, x2, x3] **implies it** = x2.

func x_3 → Element **of** X3 **means** x = [x1, x2, x3] **implies it** = x3.

Theorem MCART_1:47. X1 ≠ ∅ & X2 ≠ ∅ & X3 ≠ ∅ **implies for** x **being** Element **of** ⟦X1, X2, X3⟧ **for** x1, x2, x3 **st** x = [x1, x2, x3] **holds** x_1 = x1 & x_2 = x2 & x_3 = x3.

Theorem MCART_1:48. X1 ≠ ∅ & X2 ≠ ∅ & X3 ≠ ∅ **implies for** x **being** Element **of** ⟦X1, X2, X3⟧ **holds** x = [x_1, x_2, x_3].

Theorem MCART_1:49. X ⊆ ⟦X, Y, Z⟧ **or** X ⊆ ⟦Y, Z, X⟧ **or** X ⊆ ⟦Z, X, Y⟧ **implies** X = ∅.

Theorem MCART_1:50. X1 ≠ ∅ & X2 ≠ ∅ & X3 ≠ ∅ **implies for** x **being** Element **of** ⟦X1, X2, X3⟧ **holds** x_1 = (x **qua** Any)$_{11}$ & x_2 = (x **qua** Any)$_{12}$ & x_3 = (x **qua** Any)$_2$.

Theorem MCART_1:51. X1 ≠ ∅ & X2 ≠ ∅ & X3 ≠ ∅ **implies for** x **being** Element **of** ⟦X1, X2, X3⟧ **holds** x ≠ x_1 & x ≠ x_2 & x ≠ x_3.

Theorem MCART_1:52. ⟦X1, X2, X3⟧ meets ⟦Y1, Y2, Y3⟧ **implies** X1 meets Y1 & X2 meets Y2 & X3 meets Y3.

Theorem MCART_1:53. ⟦X1, X2, X3, X4⟧ = ⟦⟦⟦X1, X2⟧, X3⟧, X4⟧.

Theorem MCART_1:54. ⟦⟦X1, X2⟧, X3, X4⟧ = ⟦X1, X2, X3, X4⟧.

Theorem MCART_1:55. X1 ≠ ∅ & X2 ≠ ∅ & X3 ≠ ∅ & X4 ≠ ∅ **iff** ⟦X1, X2, X3, X4⟧ ≠ ∅.

Theorem MCART_1:56. X1 ≠ ∅ & X2 ≠ ∅ & X3 ≠ ∅ & X4 ≠ ∅ **implies** (⟦X1, X2, X3, X4⟧ = ⟦Y1, Y2, Y3, Y4⟧ **implies** X1 = Y1 & X2 = Y2 & X3 = Y3 & X4 = Y4).

Theorem MCART_1:57. ⟦X1, X2, X3, X4⟧ ≠ ∅ & ⟦X1, X2, X3, X4⟧ = ⟦Y1, Y2, Y3, Y4⟧ **implies** X1 = Y1 & X2 = Y2 & X3 = Y3 & X4 = Y4.

Theorem MCART_1:58. ⟦X, X, X, X⟧ = ⟦Y, Y, Y, Y⟧ **implies** X = Y.

reserve xx4 **for** Element **of** X4.

Definition

let X1, X2, X3, X4.

assume X1 $\neq \emptyset$ & X2 $\neq \emptyset$ & X3 $\neq \emptyset$ & X4 $\neq \emptyset$.

let x **be** Element **of** $[\![$X1, X2, X3, X4$]\!]$.

func x_1 $\rightarrow$ Element **of** X1 **means** x = [x1, x2, x3, x4] **implies** **it** = x1.

func x_2 $\rightarrow$ Element **of** X2 **means** x = [x1, x2, x3, x4] **implies** **it** = x2.

func x_3 $\rightarrow$ Element **of** X3 **means** x = [x1, x2, x3, x4] **implies** **it** = x3.

func x_4 $\rightarrow$ Element **of** X4 **means** x = [x1, x2, x3, x4] **implies** **it** = x4.

Theorem MCART_1:59. X1 $\neq \emptyset$ & X2 $\neq \emptyset$ & X3 $\neq \emptyset$ & X4 $\neq \emptyset$ **implies** **for** x **being** Element **of** $[\![$X1, X2, X3, X4$]\!]$ **for** x1, x2, x3, x4 **st** x = [x1, x2, x3, x4] **holds** x_1 = x1 & x_2 = x2 & x_3 = x3 & x_4 = x4.

Theorem MCART_1:60. X1 $\neq \emptyset$ & X2 $\neq \emptyset$ & X3 $\neq \emptyset$ & X4 $\neq \emptyset$ **implies** **for** x **being** Element **of** $[\![$X1, X2, X3, X4$]\!]$ **holds** x = [x_1, x_2, x_3, x_4].

Theorem MCART_1:61. X1 $\neq \emptyset$ & X2 $\neq \emptyset$ & X3 $\neq \emptyset$ & X4 $\neq \emptyset$ **implies** **for** x **being** Element **of** $[\![$X1, X2, X3, X4$]\!]$ **holds** x_1 = (x **qua** Any)$_{111}$ & x_2 = (x **qua** Any)$_{112}$ & x_3 = (x **qua** Any)$_{12}$ & x_4 = (x **qua** Any)$_2$.

Theorem MCART_1:62. X1 $\neq \emptyset$ & X2 $\neq \emptyset$ & X3 $\neq \emptyset$ & X4 $\neq \emptyset$ **implies** **for** x **being** Element **of** $[\![$X1, X2, X3, X4$]\!]$ **holds** x $\neq$ x_1 & x $\neq$ x_2 & x $\neq$ x_3 & x $\neq$ x_4.

Theorem MCART_1:63. X1 $\subseteq$ $[\![$X1, X2, X3, X4$]\!]$ **or** X1 $\subseteq$ $[\![$X2, X3, X4, X1$]\!]$ **or** X1 $\subseteq$ $[\![$X3, X4, X1, X2$]\!]$ **or** X1 $\subseteq$ $[\![$X4, X1, X2, X3$]\!]$ **implies** X1 = $\emptyset$.

Theorem MCART_1:64. $[\![$X1, X2, X3, X4$]\!]$ meets $[\![$Y1, Y2, Y3, Y4$]\!]$ **implies** X1 meets Y1 & X2 meets Y2 & X3 meets Y3 & X4 meets Y4.

Theorem MCART_1:65. $[\![$\{x1\}, \{x2\}, \{x3\}, \{x4\}$]\!]$ = \{[x1, x2, x3, x4]\}.

Theorem MCART_1:66. $[\![$X, Y$]\!]$ $\neq \emptyset$ **implies** **for** x **being** Element **of** $[\![$X, Y$]\!]$ **holds** x $\neq$ x_1 & x $\neq$ x_2.

Theorem MCART_1:67. x $\in$ $[\![$X, Y$]\!]$ **implies** x $\neq$ x_1 & x $\neq$ x_2.

reserve A1 **for** (Subset **of** X1), A2 **for** (Subset **of** X2), A3 **for** (Subset **of** X3), A4 **for** Subset **of** X4.

reserve x **for** Element **of** $[\![$X1, X2, X3$]\!]$.

Theorem MCART_1:68. X1 $\neq \emptyset$ & X2 $\neq \emptyset$ & X3 $\neq \emptyset$ **implies** **for** x1, x2, x3 **st** x = [x1, x2, x3] **holds** x_1 = x1 & x_2 = x2 & x_3 = x3.

Theorem MCART_1:69. X1 $\neq \emptyset$ & X2 $\neq \emptyset$ & X3 $\neq \emptyset$ & (**for** xx1, xx2, xx3 **st** x = [xx1, xx2, xx3] **holds** y1 = xx1) **implies** y1 = x_1.

Theorem MCART_1:70. X1 $\neq \emptyset$ & X2 $\neq \emptyset$ & X3 $\neq \emptyset$ & (**for** xx1, xx2, xx3 **st** x = [xx1, xx2, xx3] **holds** y2 = xx2) **implies** y2 = x_2.

Theorem MCART_1:71. X1 $\neq \emptyset$ & X2 $\neq \emptyset$ & X3 $\neq \emptyset$ & (**for** xx1, xx2, xx3 **st** x = [xx1, xx2, xx3] **holds** y3 = xx3) **implies** y3 = x_3.

Theorem MCART_1:72. z $\in$ $[\![$X1, X2, X3$]\!]$ **implies** **ex** x1, x2, x3 **st** x1 $\in$ X1 & x2 $\in$ X2 & x3 $\in$ X3 & z = [x1, x2, x3].

Theorem MCART_1:73. [x1, x2, x3] $\in$ $[\![$X1, X2, X3$]\!]$ **iff** x1 $\in$ X1 & x2 $\in$ X2 & x3 $\in$ X3.

Theorem MCART_1:74. (**for** z **holds** z $\in$ Z **iff ex** x1, x2, x3 **st** x1 $\in$ X1 & x2 $\in$ X2 & x3 $\in$ X3 & z = [x1, x2, x3]) **implies** Z = $[\![$X1, X2, X3$]\!]$.

Theorem MCART_1:75. X1 $\neq \emptyset$ & X2 $\neq \emptyset$ & X3 $\neq \emptyset$ & Y1 $\neq \emptyset$ & Y2 $\neq \emptyset$ & Y3 $\neq \emptyset$ **implies for** x **being** (Element **of** $[\![$X1, X2, X3$]\!]$), y **being** Element **of** $[\![$Y1, Y2, Y3$]\!]$ **holds** x = y **implies** $x_1 = y_1$ & $x_2 = y_2$ & $x_3 = y_3$.

Theorem MCART_1:76. **for** x **being** Element **of** $[\![$X1, X2, X3$]\!]$ **st** x $\in$ $[\![$A1, A2, A3$]\!]$ **holds** $x_1 \in$ A1 & $x_2 \in$ A2 & $x_3 \in$ A3.

Theorem MCART_1:77. X1 $\subseteq$ Y1 & X2 $\subseteq$ Y2 & X3 $\subseteq$ Y3 **implies** $[\![$X1, X2, X3$]\!]$ $\subseteq$ $[\![$Y1, Y2, Y3$]\!]$.

reserve x **for** Element **of** $[\![$X1, X2, X3, X4$]\!]$.

Theorem MCART_1:78. X1 $\neq \emptyset$ & X2 $\neq \emptyset$ & X3 $\neq \emptyset$ & X4 $\neq \emptyset$ **implies for** x1, x2, x3, x4 **st** x = [x1, x2, x3, x4] **holds** x_1 = x1 & x_2 = x2 & x_3 = x3 & x_4 = x4.

Theorem MCART_1:79. X1 $\neq \emptyset$ & X2 $\neq \emptyset$ & X3 $\neq \emptyset$ & X4 $\neq \emptyset$ & (**for** xx1, xx2, xx3, xx4 **st** x = [xx1, xx2, xx3, xx4] **holds** y1 = xx1) **implies** y1 = x_1.

Theorem MCART_1:80. X1 $\neq \emptyset$ & X2 $\neq \emptyset$ & X3 $\neq \emptyset$ & X4 $\neq \emptyset$ & (**for** xx1, xx2, xx3, xx4 **st** x = [xx1, xx2, xx3, xx4] **holds** y2 = xx2) **implies** y2 = x_2.

Theorem MCART_1:81. X1 $\neq \emptyset$ & X2 $\neq \emptyset$ & X3 $\neq \emptyset$ & X4 $\neq \emptyset$ & (**for** xx1, xx2, xx3, xx4 **st** x = [xx1, xx2, xx3, xx4] **holds** y3 = xx3) **implies** y3 = x_3.

Theorem MCART_1:82. X1 $\neq \emptyset$ & X2 $\neq \emptyset$ & X3 $\neq \emptyset$ & X4 $\neq \emptyset$ & (**for** xx1, xx2, xx3, xx4 **st** x = [xx1, xx2, xx3, xx4] **holds** y4 = xx4) **implies** y4 = x_4.

Theorem MCART_1:83. z $\in$ $[\![$X1, X2, X3, X4$]\!]$ **implies ex** x1, x2, x3, x4 **st** x1 $\in$ X1 & x2 $\in$ X2 & x3 $\in$ X3 & x4 $\in$ X4 & z = [x1, x2, x3, x4].

Theorem MCART_1:84. [x1, x2, x3, x4] $\in$ $[\![$X1, X2, X3, X4$]\!]$ **iff** x1 $\in$ X1 & x2 $\in$ X2 & x3 $\in$ X3 & x4 $\in$ X4.

Theorem MCART_1:85. (**for** z **holds** z $\in$ Z **iff ex** x1, x2, x3, x4 **st** x1 $\in$ X1 & x2 $\in$ X2 & x3 $\in$ X3 & x4 $\in$ X4 & z = [x1, x2, x3, x4]) **implies** Z = $[\![$X1, X2, X3, X4$]\!]$.

Theorem MCART_1:86. X1 $\neq \emptyset$ & X2 $\neq \emptyset$ & X3 $\neq \emptyset$ & X4 $\neq \emptyset$ & Y1 $\neq \emptyset$ & Y2 $\neq \emptyset$ & Y3 $\neq \emptyset$ & Y4 $\neq \emptyset$ **implies for** x **being** (Element **of** $[\![$X1, X2, X3, X4$]\!]$), y **being** Element **of** $[\![$Y1, Y2, Y3, Y4$]\!]$ **holds** x = y **implies** $x_1 = y_1$ & $x_2 = y_2$ & $x_3 = y_3$ & $x_4 = y_4$.

Theorem MCART_1:87. **for** x **being** Element **of** $[\![$X1, X2, X3, X4$]\!]$ **st** x $\in$ $[\![$A1, A2, A3, A4$]\!]$ **holds** $x_1 \in$ A1 & $x_2 \in$ A2 & $x_3 \in$ A3 & $x_4 \in$ A4.

Theorem MCART_1:88. X1 $\subseteq$ Y1 & X2 $\subseteq$ Y2 & X3 $\subseteq$ Y3 & X4 $\subseteq$ Y4 **implies** $[\![$X1, X2, X3, X4$]\!]$ $\subseteq$ $[\![$Y1, Y2, Y3, Y4$]\!]$.

Definition

let X1, X2, A1, A2.

redefine

func $[\![A1, A2]\!] \to$ Subset **of** $[\![X1, X2]\!]$.

Definition

let X1, X2, X3, A1, A2, A3.

redefine

func $[\![A1, A2, A3]\!] \to$ Subset **of** $[\![X1, X2, X3]\!]$.

Definition

let X1, X2, X3, X4, A1, A2, A3, A4.

redefine

func $[\![A1, A2, A3, A4]\!] \to$ Subset **of** $[\![X1, X2, X3, X4]\!]$.

Chapter 19

REAL_1

Basic Properties of Real Numbers

by

Krzysztof Hryniewiecki [1]

Warsaw University

Summary. Basic facts of arithmetics of real numbers are presented: definitions and properties of the complement element, the inverse element, subtraction and division; some basic properties of the set REAL (e.g. density), and the scheme of separation for sets of reals.

The symbols used in this article are introduced in vocabularies REAL_1 and BOOLE. The articles TARSKI and BOOLE provide the terminology and notation for this article.

reserve x, y, z, t **for** Real.

reserve a, b, c, d **for** Element **of** REAL.

reserve r **for** Any.

Definition

let x, y.

redefine

func x+y → Real.

func x·y → Real.

Theorem REAL_1:1. r **is** Real **iff** r ∈ REAL.

[1]Supported by RPBP.III-24.C1.

Theorem REAL_1:2. x+y = y+x.

Theorem REAL_1:3. x+(y+z) = (x+y)+z.

Theorem REAL_1:4. x+0 = x & 0+x = x.

Theorem REAL_1:5. x·y = y·x.

Theorem REAL_1:6. x·(y·z) = (x·y)·z.

Theorem REAL_1:7. x·1 = x & 1·x = x.

Theorem REAL_1:8. (x+y)·z = x·z+y·z & z·(x+y) = z·x+z·y.

Theorem REAL_1:9. (z ≠ 0 & x ≠ y) **implies** (x·z ≠ y·z & z·x ≠ y·z & z·x ≠ z·y & x·z ≠ z·y).

Theorem REAL_1:10. (z+x = z+y **or** x+z = y+z **or** z+x = y+z **or** x+z = z+y) **implies** x = y.

Theorem REAL_1:11. x ≠ y **iff** x+z ≠ y+z.

Theorem REAL_1:12. (z ≠ 0 & (x·z = y·z **or** z·x = z·y **or** x·z = z·y **or** z·x = y·z)) **implies** x = y.

Definition

let x.

func−x → Real **means** x+**it** = 0.

assume x ≠ 0.

func x^{-1} → Real **means** x·**it** = 1.

Definition

let x, y.

func x−y → Real **means** **it** = x+(−y).

assume y ≠ 0.

func x/y → Real **means** **it** = $x \cdot y^{-1}$.

Theorem REAL_1:13. x+−x = 0 &−x+x = 0.

Theorem REAL_1:14. x−y = x+−y.

Theorem REAL_1:15. x ≠ 0 **implies** $x \cdot x^{-1}$ = 1 & $x^{-1} \cdot x$ = 1.

Theorem REAL_1:16. y ≠ 0 **implies** (x/y = $x \cdot y^{-1}$ & x/y = $y^{-1} \cdot x$).

Theorem REAL_1:17. x+y−z = x+(y−z).

Theorem REAL_1:18. −(−x) = x.

Theorem REAL_1:19. 0−x =−x.

Theorem REAL_1:20. x·0 = 0 & 0·x = 0.

Theorem REAL_1:21. (−x)·y =−(x·y) & x·(−y) =−(x·y) & (−x)·y = x·(−y).

Theorem REAL_1:22. x ≠ 0 **iff**−x ≠ 0.

Theorem REAL_1:23. x·y = 0 **iff** (x = 0 **or** y = 0).

Theorem REAL_1:24. x $\neq$ 0 & y $\neq$ 0 **implies** $x^{-1} \cdot y^{-1} = (x \cdot y)^{-1}$.

Theorem REAL_1:25. x−0 = x.

Theorem REAL_1:26. −0 = 0.

Theorem REAL_1:27. x−(y+z) = x−y−z.

Theorem REAL_1:28. x−(y−z) = x−y+z.

Theorem REAL_1:29. x·(y−z) = x·y−x·z & (y−z)·x = y·x−z·x.

Theorem REAL_1:30. x+z = y **implies** (x = y−z & z = y−x).

Theorem REAL_1:31. x $\neq$ 0 **implies** $x^{-1} \neq 0$.

Theorem REAL_1:32. x $\neq$ 0 **implies** $x^{-1-1} = x$.

Theorem REAL_1:33. x $\neq$ 0 **implies** ($1/x = x^{-1}$ & $1/x^{-1} = x$).

Theorem REAL_1:34. x $\neq$ 0 **implies** x·(1/x) = 1 & (1/x)·x = 1.

Theorem REAL_1:35. (y $\neq$ 0 & t $\neq$ 0) **implies** (x/y)·(z/t) = (x·z)/(y·t).

Theorem REAL_1:36. x−x = 0.

Theorem REAL_1:37. x $\neq$ 0 **implies** x/x = 1.

Theorem REAL_1:38. y $\neq$ 0 & z $\neq$ 0 **implies** x/y = (x·z)/(y·z).

Theorem REAL_1:39. y $\neq$ 0 **implies** (−x/y = (−x)/y & x/(−y) =−x/y).

Theorem REAL_1:40. z $\neq$ 0 **implies** (x/z+y/z = (x+y)/z) & (x/z−y/z = (x−y)/z).

Theorem REAL_1:41. y $\neq$ 0 & t $\neq$ 0 **implies** (x/y+z/t = (x·t+z·y)/(y·t)) & (x/y−z/t = (x·t−z·y)/(y·t)).

Theorem REAL_1:42. y $\neq$ 0 & z $\neq$ 0 **implies** x/(y/z) = (x·z)/y.

Theorem REAL_1:43. y $\neq$ 0 **implies** x/y·y = x.

Theorem REAL_1:44. **for** x, y **ex** z **st** (x = y+z & x = z+y).

Theorem REAL_1:45. **for** x, y **st** y $\neq$ 0 **ex** z **st** (x = y·z & x = z·y).

Theorem REAL_1:46. x $\leqslant$ y & y $\leqslant$ x **implies** x = y.

Theorem REAL_1:47. x $\leqslant$ y & y $\leqslant$ z **implies** x $\leqslant$ z.

Theorem REAL_1:48. x $\leqslant$ y **or** y $\leqslant$ x.

Theorem REAL_1:49. x $\leqslant$ y **implies** (x+z $\leqslant$ y+z & x−z $\leqslant$ y−z).

Theorem REAL_1:50. x $\leqslant$ y **iff**−y $\leqslant$−x.

Theorem REAL_1:51. x $\leqslant$ y & 0 $\leqslant$ z **implies** (x·z $\leqslant$ y·z & z·x $\leqslant$ z·y & z·x $\leqslant$ y·z & x·z $\leqslant$ z·y).

Theorem REAL_1:52. x $\leqslant$ y & z $\leqslant$ 0 **implies** (y·z $\leqslant$ x·z & z·y $\leqslant$ z·x & y·z $\leqslant$ z·x & z·y $\leqslant$ x·z).

Theorem REAL_1:53. x $\leqslant$ y **iff** x+z $\leqslant$ y+z.

Theorem REAL_1:54. x $\leqslant$ y **iff** x−z $\leqslant$ y−z.

Theorem REAL_1:55. (x ⩽ y & z ⩽ t) **implies** (x+z ⩽ y+t & x+z ⩽ t+y & z+x ⩽ t+y & z+x ⩽ y+t).

Theorem REAL_1:56. x ⩽ x.

Definition

let x, y.

pred x < y **means** x ⩽ y & x ≠ y.

Theorem REAL_1:57. x < y **iff** (x ⩽ y & x ≠ y).

Theorem REAL_1:58. ((x ⩽ y & y < z) **or** (x < y & y ⩽ z) **or** (x < y & y < z)) **implies** x < z.

Theorem REAL_1:59. x < y **implies** (x+z < y+z & x−z < y−z & z+x < z+y & x+z < z+y & z+x < y+z).

Theorem REAL_1:60. (x+z < y+z **or** z+x < z+y **or** x+z < z+y **or** z+x < y+z **or** x−z < y−z) **implies** x < y.

Theorem REAL_1:61. x ≠ y **implies** x < y **or** y < x.

Theorem REAL_1:62. **not** x < y **iff** y ⩽ x.

Theorem REAL_1:63. x < y **or** y < x **or** x = y.

Theorem REAL_1:64. x < y **implies** **not** y < x.

Theorem REAL_1:65. 0 < 1.

Theorem REAL_1:66. x < 0 **iff** 0 <−x.

Theorem REAL_1:67. ((x < y & z ⩽ t) **or** (x ⩽ y & z < t) **or** (x < y & z < t)) **implies** (x+z < y+t & z+x < y+t & z+x < t+y & x+z < t+y).

Theorem REAL_1:68. x < y **iff**−y <−x.

Theorem REAL_1:69. **for** x, y **st** 0 < x **holds** y < y+x.

Theorem REAL_1:70. 0 < z & x < y **implies** (x·z < y·z & z·x < z·y & x·z < z·y & z·x < y·z).

Theorem REAL_1:71. z < 0 & x < y **implies** (y·z < x·z & z·y < z·x & y·z < z·x & z·y < x·z).

Theorem REAL_1:72. 0 < z **implies** 0 < z^{-1}.

Theorem REAL_1:73. 0 < z **implies** (x < y **iff** x/z < y/z).

Theorem REAL_1:74. z < 0 **implies** (x < y **iff** y/z < x/z).

Theorem REAL_1:75. x < y **implies** **ex** z **st** x < z & z < y.

Theorem REAL_1:76. **for** x **ex** y **st** x < y.

Theorem REAL_1:77. **for** x **ex** y **st** y < x.

Theorem REAL_1:78. **for** X, Y **being** Subset **of** REAL **st** (**ex** x **st** x ∈ X) & (**ex** x **st** x ∈ Y) & **for** x, y **st** x ∈ X & y ∈ Y **holds** x ⩽ y **ex** z **st** **for** x, y **st** x ∈ X & y ∈ Y **holds** x ⩽ z & z ⩽ y.

scheme SepReal{P[Real]}: **ex** X **being set of** Real **st for** x **holds** x $\in$ X **iff** P[x].

Theorem REAL_1:79. y =−x **iff** x+y = 0.

Theorem REAL_1:80. **for** x, y **st** x $\neq$ 0 **holds** y = x^{-1} **iff** x·y = 1.

Theorem REAL_1:81. **for** x, y **st** x $\neq$ 0 & y $\neq$ 0 **holds** $(x/y)^{-1}$ = y/x.

Theorem REAL_1:82. **for** x, y, z, t **st** y $\neq$ 0 & z $\neq$ 0 & t $\neq$ 0 **holds** (x/y)/(z/t) = (x·t)/(y·z).

Theorem REAL_1:83. −(x−y) = y−x.

Theorem REAL_1:84. (x+y $\leqslant$ z **iff** x $\leqslant$ z−y).

Theorem REAL_1:85. (x+y $\leqslant$ z **iff** y $\leqslant$ z−x).

Theorem REAL_1:86. (x $\leqslant$ y+z **iff** x−y $\leqslant$ z).

Theorem REAL_1:87. (x $\leqslant$ y+z **iff** x−z $\leqslant$ y).

Theorem REAL_1:88. (x+y < z **iff** x < z−y).

Theorem REAL_1:89. (x+y < z **iff** y < z−x).

Theorem REAL_1:90. (x < z+y **iff** x−z < y).

Theorem REAL_1:91. (x < y+z **iff** x−z < y).

Theorem REAL_1:92. ((x $\leqslant$ y & z $\leqslant$ t) **implies** x−t $\leqslant$ y−z) & (((x < y & z $\leqslant$ t) **or** (x $\leqslant$ y & z < t) **or** (x < y & z < t)) **implies** x−t < y−z).

Theorem REAL_1:93. 0 $\leqslant$ x·x.

Chapter 20

ORDINAL1

The Ordinal Numbers

Transfinite Induction and Defining by Transfinite Induction

by

Grzegorz Bancerek [1]

Warsaw University (Białystok)

Summary. We introduce some consequences of the regularity axiom, the successor of a set, ∈-transitivity and ∈-connectedness, the definition and basic properties of ordinal numbers and sets of ordinals, transfinite sequences, transfinite induction, and schemes of defining by transfinite induction.

The symbols used in this article are introduced in the following vocabularies: BOOLE, FAM_OP, REAL_1, FUNC_REL, FUNC, and ORDINAL. The terminology and notation used in this article have been introduced in the following articles: TARSKI, BOOLE, ENUMSET1, and FUNCT_1.

reserve X, Y, Z, A, B, C, X1, X2, X3, X4, X5, X6 **for** set, x, y, z, a, b, c **for** Any.

Theorem ORDINAL1:1. **not** X ∈ X.

Theorem ORDINAL1:2. **not** (X ∈ Y & Y ∈ X).

Theorem ORDINAL1:3. **not** (X ∈ Y & Y ∈ Z & Z ∈ X).

Theorem ORDINAL1:4. **not** (X1 ∈ X2 & X2 ∈ X3 & X3 ∈ X4 & X4 ∈ X1).

Theorem ORDINAL1:5. **not** (X1 ∈ X2 & X2 ∈ X3 & X3 ∈ X4 & X4 ∈ X5 & X5 ∈ X1).

[1] Supported by RPBP.III-24.C1.

Theorem ORDINAL1:6. **not** (X1 ∈ X2 & X2 ∈ X3 & X3 ∈ X4 & X4 ∈ X5 & X5 ∈ X6 & X6 ∈ X1).

Theorem ORDINAL1:7. Y ∈ X **implies not** X ⊆ Y.

scheme Comprehension{A() → set, P[set]}: **ex** B **st for** Z **being** set **holds** Z ∈ B **iff** Z ∈ A() & P[Z].

Theorem ORDINAL1:8. (**for** X **holds** X ∈ A **iff** X ∈ B) **implies** A = B.

Definition

let X.

func succ X → set **means it** = X∪{X}.

Theorem ORDINAL1:9. succ X = X∪{X}.

Theorem ORDINAL1:10. X ∈ succ X.

Theorem ORDINAL1:11. succ X ≠ ∅.

Theorem ORDINAL1:12. succ X = succ Y **implies** X = Y.

Theorem ORDINAL1:13. x ∈ succ X **iff** x ∈ X **or** x = X.

Theorem ORDINAL1:14. X ≠ succ X.

reserve a, b, c, d **for** Any, X, Y, Z, x, y, z **for** set.

Definition

let X.

pred X is ∈-transitive **means for** x **st** x ∈ X **holds** x ⊆ X.

pred X is ∈-connected **means for** x, y **st** x ∈ X & y ∈ X **holds** x ∈ y **or** x = y **or** y ∈ x.

Theorem ORDINAL1:15. X is ∈-transitive **iff for** x **st** x ∈ X **holds** x ⊆ X.

Theorem ORDINAL1:16. X is ∈-connected **iff for** x, y **st** x ∈ X & y ∈ X **holds** x ∈ y **or** x = y **or** y ∈ x.

Definition

mode Ordinal → set **means it** is ∈-transitive & **it** is ∈-connected.

reserve A, B, C, D **for** Ordinal.

Theorem ORDINAL1:17. X **is** Ordinal **iff** X is ∈-transitive & X is ∈-connected.

Theorem ORDINAL1:18. x ∈ A **implies** x ⊆ A.

Theorem ORDINAL1:19. A ∈ B & B ∈ C **implies** A ∈ C.

Theorem ORDINAL1:20. x ∈ A & y ∈ A **implies** x ∈ y **or** x = y **or** y ∈ x.

Theorem ORDINAL1:21. **for** x, A **being** Ordinal **st** x ⊆ A & x ≠ A **holds** x ∈ A.

Theorem ORDINAL1:22. A ⊆ B & B ∈ C **implies** A ∈ C.

Theorem ORDINAL1:23. a ∈ A **implies** a **is** Ordinal.

Theorem ORDINAL1:24. A ∈ B **or** A = B **or** B ∈ A.

Theorem ORDINAL1:25. A $\subseteq$ B **or** B $\subseteq$ A.

Theorem ORDINAL1:26. A $\subseteq$ B **or** B $\in$ A.

Theorem ORDINAL1:27. $\emptyset$ **is** Ordinal.

Definition

func **0** $\rightarrow$ Ordinal **means** **it** = $\emptyset$.

Theorem ORDINAL1:28. **0** = $\emptyset$.

Theorem ORDINAL1:29. x **is** Ordinal **implies** succ x **is** Ordinal.

Theorem ORDINAL1:30. x **is** Ordinal **implies** $\bigcup$x **is** Ordinal.

Definition

let A.

redefine

func succ A $\rightarrow$ Ordinal.

func $\bigcup$A $\rightarrow$ Ordinal.

Theorem ORDINAL1:31. (**for** x **st** x $\in$ X **holds** x **is** Ordinal & x $\subseteq$ X) **implies** X **is** Ordinal.

Theorem ORDINAL1:32. X $\subseteq$ A & X $\neq$ $\emptyset$ **implies** **ex** C **st** C $\in$ X & **for** B **st** B $\in$ X **holds** C $\subseteq$ B.

Theorem ORDINAL1:33. A $\in$ B **iff** succ A $\subseteq$ B.

Theorem ORDINAL1:34. A $\in$ succ C **iff** A $\subseteq$ C.

scheme Ordinal_Min{P[Ordinal]}: **ex** A **st** P[A] & **for** B **st** P[B] **holds** A $\subseteq$ B **provided** A: **ex** A **st** P[A].

scheme Transfinite_Ind{P[Ordinal]}: **for** A **holds** P[A] **provided** A: **for** A **st** **for** C **st** C $\in$ A **holds** P[C] **holds** P[A].

Theorem ORDINAL1:35. **for** X **st** **for** a **st** a $\in$ X **holds** a **is** Ordinal **holds** $\bigcup$X **is** Ordinal.

Theorem ORDINAL1:36. **for** X **st** **for** a **st** a $\in$ X **holds** a **is** Ordinal **ex** A **st** X $\subseteq$ A.

Theorem ORDINAL1:37. **not** **ex** X **st** **for** x **holds** x $\in$ X **iff** x **is** Ordinal.

Theorem ORDINAL1:38. **not** **ex** X **st** **for** A **holds** A $\in$ X.

Theorem ORDINAL1:39. **for** X **ex** A **st** **not** A $\in$ X & **for** B **st** **not** B $\in$ X **holds** A $\subseteq$ B.

Definition

let A.

pred A is limit ordinal **means** A = $\bigcup$A.

Theorem ORDINAL1:40. A is limit ordinal **iff** A = $\bigcup$A.

Theorem ORDINAL1:41. **for** A **holds** A is limit ordinal **iff** **for** C **st** C $\in$ A **holds** succ C $\in$ A.

Theorem ORDINAL1:42. **not** A is limit ordinal **iff ex** B **st** A = succ B.

reserve F, G, H **for** Function.

Definition

mode transfinite sequence → Function **means ex** A **st** dom **it** = A.

Definition

let Z.

mode transfinite sequence **of** Z → transfinite sequence **means** rng **it** ⊆ Z.

Theorem ORDINAL1:43. F **is** transfinite sequence **iff ex** A **st** dom F = A.

Theorem ORDINAL1:44. F **is** transfinite sequence **of** Z **iff** F **is** transfinite sequence & rng F ⊆ Z.

Theorem ORDINAL1:45. ∅ **is** transfinite sequence **of** Z.

reserve L, L1, L2 **for** transfinite sequence.

Theorem ORDINAL1:46. dom F **is** Ordinal **implies** F **is** transfinite sequence **of** rng F.

Definition

let L.

redefine

func dom L → Ordinal.

Theorem ORDINAL1:47. X ⊆ Y **implies for** L **being** transfinite sequence **of** X **holds** L **is** transfinite sequence **of** Y.

Definition

let L, A.

redefine

func L↾A → transfinite sequence **of** rng L.

Theorem ORDINAL1:48. **for** L **being** transfinite sequence **of** X **for** A **holds** L↾A **is** transfinite sequence **of** X.

Theorem ORDINAL1:49. (**for** a **st** a ∈ X **holds** a **is** transfinite sequence) & (**for** L1, L2 **st** L1 ∈ X & L2 ∈ X **holds** graph L1 ⊆ graph L2 **or** graph L2 ⊆ graph L1) **implies** ⋃X **is** transfinite sequence.

scheme TS_Uniq{A() → Ordinal, H(transfinite sequence) → Any, L1() → transfinite sequence, L2() → transfinite sequence}: L1() = L2() **provided** B: dom L1() = A() & **for** B, L **st** B ∈ A() & L = L1()↾B **holds** L1().B = H(L) **and** C: dom L2() = A() & **for** B, L **st** B ∈ A() & L = L2()↾B **holds** L2().B = H(L).

scheme TS_Exist{A() → Ordinal, H(transfinite sequence) → Any}: **ex** L **st** dom L = A() & **for** B, L1 **st** B ∈ A() & L1 = L↾B **holds** L.B = H(L1).

scheme Func_TS{L() → transfinite sequence, F(Ordinal) → Any, H(transfinite sequence) → Any}: **for** B **st** B ∈ dom L() **holds** L().B = H(L()↾B) **provided** A: **for** A, a **holds** a

= F(A) **iff** **ex** L **st** a = H(L) & dom L = A & **for** B **st** B $\in$ A **holds** L.B = H(L$\restriction$B) **and** B: **for** A **st** A $\in$ dom L() **holds** L().A = F(A).

Chapter 21

NAT_1

The Fundamental Properties of Natural Numbers

by

Grzegorz Bancerek [1]

Warsaw University (Białystok)

Summary. Some fundamental properties of addition, multiplication, order relations, exact division, the remainder, divisibility, the least common multiple, the greatest common divisor are presented. A proof of Euclid algorithm is also given.

The symbols used in this article are introduced in the following vocabularies: BOOLE, REAL_1, and NAT_1. The terminology and notation used in this article have been introduced in the following articles: TARSKI, BOOLE, and REAL_1.

reserve x, y, z **for** Real, k, l, m, n, u, w, v **for** Nat, X, Y, Z **for set of** Real.

Theorem NAT_1:1. x **is** Nat **implies** x+1 **is** Nat.

Theorem NAT_1:2. **for** X **st** 0 ∈ X & **for** x **st** x ∈ X **holds** x+1 ∈ X **for** k **holds** k ∈ X.

Theorem NAT_1:3. k+n = n+k.

Theorem NAT_1:4. k+m+n = k+(m+n).

Theorem NAT_1:5. k+0 = k & 0+k = k.

Theorem NAT_1:6. k·n = n·k.

[1]Supported by RPBP.III-24.C1.

Theorem NAT_1:7. k·(m·n) = (k·m)·n.

Theorem NAT_1:8. k·1 = k & 1·k = k.

Theorem NAT_1:9. k·(n+m) = k·n+k·m & (n+m)·k = n·k+m·k.

Theorem NAT_1:10. k+m = n+m **or** k+m = m+n **or** m+k = m+n **implies** k = n.

Theorem NAT_1:11. k·0 = 0 & 0·k = 0.

Definition

let n, k.

redefine

func n+k → Nat.

scheme Ind{P[Nat]}: **for** k **holds** P[k] **provided** A: P[0] **and** B: **for** k **st** P[k] **holds** P[k+1].

Definition

let n, k.

redefine

func n·k → Nat.

Theorem NAT_1:12. k ⩽ n & n ⩽ k **implies** k = n.

Theorem NAT_1:13. k ⩽ n & n ⩽ m **implies** k ⩽ m.

Theorem NAT_1:14. k ⩽ n **or** n ⩽ k.

Theorem NAT_1:15. k ⩽ k.

Theorem NAT_1:16. k ⩽ n **implies** k+m ⩽ n+m & k+m ⩽ m+n & m+k ⩽ m+n & m+k ⩽ n+m.

Theorem NAT_1:17. k+m ⩽ n+m **or** k+m ⩽ m+n **or** m+k ⩽ m+n **or** m+k ⩽ n+m **implies** k ⩽ n.

Theorem NAT_1:18. **for** k **holds** 0 ⩽ k.

Theorem NAT_1:19. 0 ≠ k **implies** 0 < k.

Theorem NAT_1:20. k ⩽ n **implies** k·m ⩽ n·m & k·m ⩽ m·n & m·k ⩽ n·m & m·k ⩽ m·n.

Theorem NAT_1:21. 0 ≠ k+1.

Theorem NAT_1:22. k = 0 **or** **ex** n **st** k = n+1.

Theorem NAT_1:23. k+n = 0 **implies** k = 0 & n = 0.

Theorem NAT_1:24. k ≠ 0 & (n·k = m·k **or** n·k = k·m **or** k·n = k·m) **implies** n = m.

Theorem NAT_1:25. k·n = 0 **implies** k = 0 **or** n = 0.

scheme Def_by_Ind{N() → Nat, F(Nat, Nat) → Nat, P[Nat, Nat]}: (**for** k **ex** n **st** P[k, n]) & **for** k, n, m **st** P[k, n] & P[k, m] **holds** n = m **provided** A: **for** k, n **holds** P[k, n] **iff** k = 0 & n = N() **or** **ex** m, l **st** k = m+1 & P[m, l] & n = F(k, l).

Theorem NAT_1:26. **for** k, n **st** k ⩽ n+1 **holds** k ⩽ n **or** k = n+1.

Theorem NAT_1:27. **for** n, k **st** n ⩽ k & k ⩽ n+1 **holds** n = k **or** k = n+1.

Theorem NAT_1:28. **for** k, n **st** k ⩽ n **ex** m **st** n = k+m.

Theorem NAT_1:29. n = k+m **implies** k ⩽ n.

Theorem NAT_1:30. k < n **iff** k ⩽ n & k ≠ n.

Theorem NAT_1:31. **not** k < 0.

scheme Comp_Ind{P[Nat]}: **for** k **holds** P[k] **provided** A: **for** k **st** **for** n **st** n < k **holds** P[n] **holds** P[k].

scheme Min{P[Nat]}: **ex** k **st** P[k] & **for** n **st** P[n] **holds** k ⩽ n **provided** A: **ex** k **st** P[k].

scheme Max{P[Nat], N() → Nat}: **ex** k **st** P[k] & **for** n **st** P[n] **holds** n ⩽ k **provided** A: **for** k **st** P[k] **holds** k ⩽ N() **and** B: **ex** k **st** P[k].

Theorem NAT_1:32. **not** (k < n & n < k).

Theorem NAT_1:33. k < n & n < m **implies** k < m.

Theorem NAT_1:34. k < n **or** k = n **or** n < k.

Theorem NAT_1:35. **not** k < k.

Theorem NAT_1:36. k < n **implies** k+m < n+m & k+m < m+n & m+k < m+n & m+k < n+m.

Theorem NAT_1:37. k ⩽ n **implies** k ⩽ n+m.

Theorem NAT_1:38. k < n+1 **iff** k ⩽ n.

Theorem NAT_1:39. k ⩽ n & n < m **or** k < n & n ⩽ m **or** k < n & n < m **implies** k < m.

Theorem NAT_1:40. k·n = 1 **implies** k = 1 & n = 1.

Theorem NAT_1:41. k+1 ⩽ n **iff** k < n.

scheme Regr{P[Nat]}: P[0] **provided** A: **ex** k **st** P[k] **and** B: **for** k **st** k ≠ 0 & P[k] **ex** n **st** n < k & P[n].

reserve k1, t, t1 **for** Nat.

Theorem NAT_1:42. **for** m **st** 0 < m **for** n **ex** k, t **st** n = (m·k)+t & t < m.

Theorem NAT_1:43. **for** n, m, k, k1, t, t1 **st** n = m·k+t & t < m & n = m·k1+t1 & t1 < m **holds** k = k1 & t = t1.

Definition

let k, l **be** Nat.

func k÷l → Nat **means** (**ex** t **st** k = l·**it**+t & t < l) **or** **it** = 0 & l = 0.

func k mod l → Nat **means** (**ex** t **st** k = l·t+**it** & **it** < l) **or** **it** = 0 & l = 0.

Theorem NAT_1:44. **for** k, l, n **being** Nat **holds** n = k÷l **iff** (**ex** t **st** k = l·n+t & t < l) **or** n = 0 & l = 0.

Theorem NAT_1:45. **for** k, l, n **being** Nat **holds** n = k mod l **iff** (**ex** t **st** k = l·t+n & n < l) **or** n = 0 & l = 0.

Theorem NAT_1:46. **for** m, n **st** 0 < m **holds** n mod m < m.

Theorem NAT_1:47. **for** n, m **st** 0 < m **holds** n = m·(n÷m)+(n mod m).

Definition

let k, l **be** Nat.

pred k | l **means** **ex** t **st** l = k·t.

Theorem NAT_1:48. **for** k, l **being** Nat **holds** k | l **iff** **ex** t **st** l = k·t.

Theorem NAT_1:49. **for** n, m **holds** m | n **iff** n = m·(n÷m).

Theorem NAT_1:50. **for** n **holds** n | n.

Theorem NAT_1:51. **for** n, m, l **st** n | m & m | l **holds** n | l.

Theorem NAT_1:52. **for** n, m **st** n | m & m | n **holds** n = m.

Theorem NAT_1:53. k | 0 & 1 | k.

Theorem NAT_1:54. **for** n, m **st** 0 < m & n | m **holds** n ⩽ m.

Theorem NAT_1:55. **for** n, m, l **st** n | m & n | l **holds** n | m+l.

Theorem NAT_1:56. n | k **implies** n | k·m.

Theorem NAT_1:57. **for** n, m, l **st** n | m & n | m+l **holds** n | l.

Theorem NAT_1:58. n | m & n | k **implies** n | m mod k.

Definition

let k, n.

func k lcm n → Nat **means** k | **it** & n | **it** & **for** m **st** k | m & n | m **holds** **it** | m.

Definition

let k, n.

func k gcd n → Nat **means** **it** | k & **it** | n & **for** m **st** m | k & m | n **holds** m | **it**.

scheme Euklides{Q(Nat) → Nat, a() → Nat, b() → Nat}: **ex** n **st** Q(n) = a() gcd b() & Q(n+1) = 0 **provided** A: 0 < b() & b() < a() **and** B: Q(0) = a() & Q(1) = b() **and** C: **for** n **holds** Q(n+2) = Q(n) mod Q(n+1).

Chapter 22

FINSEQ_1

Segments of Natural Numbers and Finite Sequences

by

Grzegorz Bancerek [1]

Warsaw University (Białystok)

Krzysztof Hryniewiecki [2]

Warsaw University

Summary. We define the notion of an initial segment of natural numbers and prove a number of their properties. Using this notion we introduce finite sequences, subsequences, the empty sequence, a sequence of a domain, and the operation of concatenation of two sequences.

The symbols used in this article are introduced in the following vocabularies: FINSEQ, FUNC_REL, FUNC, BOOLE, REAL_1, and NAT_1. The terminology and notation used in this article have been introduced in the following articles: TARSKI, BOOLE, FUNCT_1, REAL_1, and NAT_1.

reserve k, l, m, n, k1, k2 **for** Nat, X, Y, Z **for** set, x, y, z, y1, y2 **for** Any, f, g, h **for** Function.

[1]Supported by RPBP.III-24.C1.
[2]Supported by RPBP.III-24.C1.

Definition

let n.

func Seg n → **set of** Nat **means it** = {k: 1 ⩽ k & k ⩽ n}.

Theorem FINSEQ_1:1. Seg n = {k: 1 ⩽ k & k ⩽ n}.

Theorem FINSEQ_1:2. x ∈ Seg n **implies** x **is** Nat.

Theorem FINSEQ_1:3. k ∈ Seg n **iff** 1 ⩽ k & k ⩽ n.

Theorem FINSEQ_1:4. Seg 0 = ∅ & Seg 1 = {1} & Seg 2 = {1, 2}.

Theorem FINSEQ_1:5. n = 0 **or** n ∈ Seg n.

Theorem FINSEQ_1:6. n+1 ∈ Seg (n+1).

Theorem FINSEQ_1:7. n ⩽ m **iff** Seg n ⊆ Seg m.

Theorem FINSEQ_1:8. Seg n = Seg m **implies** n = m.

Theorem FINSEQ_1:9. k ⩽ n **implies** Seg k = Seg k∩Seg n & Seg k = Seg n∩Seg k.

Theorem FINSEQ_1:10. (Seg k = Seg k∩Seg n **or** Seg k = Seg n∩Seg k) **implies** k ⩽ n.

Theorem FINSEQ_1:11. Seg n∪{n+1} = Seg (n+1).

Definition

mode FinSequence → Function **means ex** n **st** dom **it** = Seg n.

reserve p, q, r, s, t, v **for** FinSequence.

Definition

let p.

func len p → Nat **means** Seg **it** = dom p.

Theorem FINSEQ_1:12. **for** f **being** Function **holds** f **is** FinSequence **iff ex** n **st** dom f = Seg n.

Theorem FINSEQ_1:13. k = len p **iff** Seg k = dom p.

Theorem FINSEQ_1:14. ∅ **is** FinSequence.

Theorem FINSEQ_1:15. (**ex** k **st** dom f ⊆ Seg k) **implies ex** p **st** graph f ⊆ graph p.

scheme SeqEx{A() → Nat, P[Any, Any]}: **ex** p **st** dom p = Seg A() & **for** k **st** k ∈ Seg A() **holds** P[k, p.k] **provided** A: **for** k, y1, y2 **st** k ∈ Seg A() & P[k, y1] & P[k, y2] **holds** y1 = y2 **and** B: **for** k **st** k ∈ Seg A() **ex** x **st** P[k, x].

scheme SeqLambda{A() → Nat, F(Any) → Any}: **ex** p **being** FinSequence **st** len p = A() & **for** k **st** k ∈ Seg A() **holds** p.k = F(k).

Theorem FINSEQ_1:16. z ∈ graph p **implies ex** k **st** (k ∈ dom p & z = [k, p.k]).

Theorem FINSEQ_1:17. X = dom p & X = dom q & (**for** k **st** k ∈ X **holds** p.k = q.k) **implies** p = q.

Theorem FINSEQ_1:18. **for** p, q **st** (len p = len q) & **for** k **st** 1 ⩽ k & k ⩽ len p **holds** p.k = q.k **holds** p = q.

Theorem FINSEQ_1:19. p↾(Seg n) **is** FinSequence.

Theorem FINSEQ_1:20. (rng p ⊆ dom f) **implies** (f·p **is** FinSequence).

Theorem FINSEQ_1:21. k ⩽ len p & q = p↾(Seg k) **implies** len q = k & dom q = Seg k.

Definition

let D **be** DOMAIN.

mode FinSequence **of** D → FinSequence **means** rng **it** ⊆ D.

reserve D, D1, D2 **for** DOMAIN.

Theorem FINSEQ_1:22. p **is** FinSequence **of** D **iff** rng p ⊆ D.

Theorem FINSEQ_1:23. **for** D, k **for** p **being** FinSequence **of** D **holds** p↾(Seg k) **is** FinSequence **of** D.

Theorem FINSEQ_1:24. **ex** p **being** FinSequence **of** D **st** len p = k.

Definition

func ε → FinSequence **means** len **it** = 0.

Theorem FINSEQ_1:25. p = ε **iff** len p = 0.

Theorem FINSEQ_1:26. p = ε **iff** dom p = ∅.

Theorem FINSEQ_1:27. p = ε **iff** rng p = ∅.

Theorem FINSEQ_1:28. graph ε = ∅.

Theorem FINSEQ_1:29. **for** D **holds** ε **is** FinSequence **of** D.

Definition

let D **be** DOMAIN.

func ε(D) → FinSequence **of** D **means** **it** = ε.

Theorem FINSEQ_1:30. p = ε(D) **iff** dom p = ∅.

Theorem FINSEQ_1:31. ε(D) = ε.

Theorem FINSEQ_1:32. p = ε(D) **iff** len p = 0.

Theorem FINSEQ_1:33. p = ε(D) **iff** rng p = ∅.

Definition

let p, q.

func p⌢q → FinSequence **means** dom **it** = Seg (len p+len q) & (**for** k **st** k ∈ dom p **holds** **it**.k = p.k) & (**for** k **st** k ∈ dom q **holds** **it**.(len p+k) = q.k).

Theorem FINSEQ_1:34. r = p⌢q **iff** (dom r = Seg (len p+len q) & (**for** k **st** k ∈ dom p **holds** r.k = p.k) & (**for** k **st** k ∈ dom q **holds** r.(len p+k) = q.k)).

Theorem FINSEQ_1:35. len (p⌢q) = len p+len q.

Theorem FINSEQ_1:36. **for** k **st** len p+1 ⩽ k & k ⩽ len p+len q **holds** (p⌢q).k = q.(k−len p).

Theorem FINSEQ_1:37. len p < k & k ⩽ len (p⌢q) **implies** (p⌢q).k = q.(k−len p).

Theorem FINSEQ_1:38. k ∈ dom (p⌢q) **implies** (k ∈ dom p **or** (**ex** n **st** n ∈ dom q & k = len p+n)).

Theorem FINSEQ_1:39. dom p ⊆ dom (p⌢q).

Theorem FINSEQ_1:40. x ∈ dom q **implies** **ex** k **st** k = x & len p+k ∈ dom (p⌢q).

Theorem FINSEQ_1:41. k ∈ dom q **implies** len p+k ∈ dom (p⌢q).

Theorem FINSEQ_1:42. rng p ⊆ rng (p⌢q).

Theorem FINSEQ_1:43. rng q ⊆ rng (p⌢q).

Theorem FINSEQ_1:44. rng (p⌢q) = rng p∪rng q.

Theorem FINSEQ_1:45. p⌢q⌢r = p⌢(q⌢r).

Theorem FINSEQ_1:46. p⌢r = q⌢r **or** r⌢p = r⌢q **implies** p = q.

Theorem FINSEQ_1:47. p⌢ε = p & ε⌢p = p.

Theorem FINSEQ_1:48. p⌢q = ε **implies** p = ε & q = ε.

Definition

let D.

let p, q **be** FinSequence **of** D.

redefine

func p⌢q → FinSequence **of** D.

Theorem FINSEQ_1:49. **for** p, q **being** FinSequence **of** D **holds** p⌢q **is** FinSequence **of** D.

Definition

let x.

func ⟨x⟩ → FinSequence **means** dom **it** = Seg 1 & **it**.1 = x.

Theorem FINSEQ_1:50. p⌢q **is** FinSequence **of** D **implies** p **is** FinSequence **of** D & q **is** FinSequence **of** D.

Definition

let x, y.

func ⟨x, y⟩ → FinSequence **means** **it** = ⟨x⟩⌢⟨y⟩.

let z.

func ⟨x, y, z⟩ → FinSequence **means** **it** = ⟨x⟩⌢⟨y⟩⌢⟨z⟩.

Theorem FINSEQ_1:51. p = ⟨x⟩ **iff** dom p = Seg 1 & p.1 = x.

Theorem FINSEQ_1:52. graph ⟨x⟩ = {[1, x]}.

Theorem FINSEQ_1:53. ⟨x, y⟩ = ⟨x⟩⌢⟨y⟩.

Theorem FINSEQ_1:54. ⟨x, y, z⟩ = ⟨x⟩⌢⟨y⟩⌢⟨z⟩.

Theorem FINSEQ_1:55. p = ⟨x⟩ **iff** dom p = Seg 1 & rng p = {x}.

Theorem FINSEQ_1:56. p = ⟨x⟩ **iff** len p = 1 & rng p = {x}.

Theorem FINSEQ_1:57. $p = \langle x \rangle$ **iff** len $p = 1$ & $p.1 = x$.

Theorem FINSEQ_1:58. $(\langle x \rangle ^\frown p).1 = x$.

Theorem FINSEQ_1:59. $(p ^\frown \langle x \rangle).(\text{len } p+1) = x$.

Theorem FINSEQ_1:60. $\langle x, y, z \rangle = \langle x \rangle ^\frown \langle y, z \rangle$ & $\langle x, y, z \rangle = \langle x, y \rangle ^\frown \langle z \rangle$.

Theorem FINSEQ_1:61. $p = \langle x, y \rangle$ **iff** len $p = 2$ & $p.1 = x$ & $p.2 = y$.

Theorem FINSEQ_1:62. $p = \langle x, y, z \rangle$ **iff** len $p = 3$ & $p.1 = x$ & $p.2 = y$ & $p.3 = z$.

Theorem FINSEQ_1:63. **for** p **st** $p \neq \varepsilon$ **holds** **ex** q, x **st** $p = q ^\frown \langle x \rangle$.

Definition

let D.

let x **be** Element **of** D.

redefine

func $\langle x \rangle \rightarrow$ FinSequence **of** D.

Definition

let D.

let S **be** SUBDOMAIN **of** D.

let x **be** Element **of** S.

redefine

func $\langle x \rangle \rightarrow$ FinSequence **of** S.

Definition

let S **be** SUBDOMAIN **of** REAL.

let x **be** Element **of** S.

redefine

func $\langle x \rangle \rightarrow$ FinSequence **of** S.

scheme IndSeq{P[FinSequence]}: **for** p **holds** P[p] **provided** A: P[ε] **and** B: **for** p, x **st** P[p] **holds** P[$p ^\frown \langle x \rangle$].

Theorem FINSEQ_1:64. **for** p, q, r, s **being** FinSequence **st** $p ^\frown q = r ^\frown s$ & len $p \leqslant$ len r **ex** t **being** FinSequence **st** $p ^\frown t = r$.

Definition

let D.

func $D^\star \rightarrow$ DOMAIN **means** $x \in$ **it** **iff** x **is** FinSequence **of** D.

Theorem FINSEQ_1:65. $x \in D^\star$ **iff** x **is** FinSequence **of** D.

Theorem FINSEQ_1:66. $\varepsilon \in D^\star$.

scheme SepSeq{D() $\rightarrow$ DOMAIN, P[FinSequence]}: **ex** X **st** (**for** x **holds** $x \in X$ **iff** **ex** p **st** ($p \in D()^\star$ & P[p] & $x = p$)).

Definition

mode FinSubsequence $\rightarrow$ Function **means** **ex** k **st** dom **it** $\subseteq$ Seg k.

Theorem FINSEQ_1:67. f **is** FinSubsequence **iff** **ex** k **st** dom f $\subseteq$ Seg k.

Theorem FINSEQ_1:68. **for** p **being** FinSequence **holds** p **is** FinSubsequence.

Theorem FINSEQ_1:69. **for** p, X **holds** (p$\restriction$X **is** FinSubsequence & X$\restriction$p **is** FinSubsequence).

reserve p′, q′ **for** FinSubsequence.

Definition

let X.

given k **such that** X $\subseteq$ Seg k.

func Sgm X $\rightarrow$ FinSequence **of** NAT **means** rng **it** = X & **for** l, m, k1, k2 **st** (1 $\leqslant$ l & l $<$ m & m $\leqslant$ len **it** & k1 = **it**.l & k2 = **it**.m) **holds** k1 $<$ k2.

Theorem FINSEQ_1:70. (**ex** k **st** X $\subseteq$ Seg k) **implies for** p **being** FinSequence **of** NAT **holds** (p = Sgm X **iff** rng p = X & **for** l, m, k1, k2 **st** (1 $\leqslant$ l & l $<$ m & m $\leqslant$ len p & k1 = p.l & k2 = p.m) **holds** k1 $<$ k2).

Theorem FINSEQ_1:71. rng Sgm dom p′ = dom p′.

Definition

let p′.

func Seq p′ $\rightarrow$ FinSequence **means it** = p′·Sgm (dom p′).

Theorem FINSEQ_1:72. **for** X **st ex** k **st** X $\subseteq$ Seg k **holds** Sgm X = ε **iff** X = $\emptyset$.

Chapter 23

FINSET_1

Finite Sets

by

Agata Darmochwał[1]

Warsaw University (Białystok)

Summary. The article contains the definition of a finite set based on the notion of finite sequence. Some theorems about properties of finite sets and finite families of sets are proved.

The symbols used in this article are introduced in the following vocabularies: FINSEQ, BOOLE, FAM_OP, COORD, FUNC, FUNC_REL, FINITE, NAT_1, REAL_1, and SFAMILY. The terminology and notation used in this article have been introduced in the following articles: TARSKI, BOOLE, ENUMSET1, SUBSET_1, FUNCT_1, ORDINAL1, MCART_1, REAL_1, NAT_1, FINSEQ_1, and SETFAM_1.

Definition

let A **be** set.

pred A is finite **means ex** p **being** FinSequence **st** rng p = A.

reserve A, B, C, D, X, Y, Y1, Y2, Z **for** set.

reserve p, q **for** FinSequence.

reserve x, y, z, x1, x2, x3, x4, x5, x6, x7, x8, y1, y2 **for** Any.

reserve f, g **for** Function.

[1]Supported by RPBP.III-24.C1.

reserve n **for** Nat.

Theorem FINSET_1:1. A is finite **iff** **ex** p **being** FinSequence **st** rng p = A.

Theorem FINSET_1:2. **for** p **being** FinSequence **holds** rng p is finite.

Theorem FINSET_1:3. Seg n is finite.

Theorem FINSET_1:4. $\emptyset$ is finite.

Theorem FINSET_1:5. {x} is finite.

Theorem FINSET_1:6. {x, y} is finite.

Theorem FINSET_1:7. {x, y, z} is finite.

Theorem FINSET_1:8. {x1, x2, x3, x4} is finite.

Theorem FINSET_1:9. {x1, x2, x3, x4, x5} is finite.

Theorem FINSET_1:10. {x1, x2, x3, x4, x5, x6} is finite.

Theorem FINSET_1:11. {x1, x2, x3, x4, x5, x6, x7} is finite.

Theorem FINSET_1:12. {x1, x2, x3, x4, x5, x6, x7, x8} is finite.

Theorem FINSET_1:13. A $\subseteq$ B & B is finite **implies** A is finite.

Theorem FINSET_1:14. A is finite & B is finite **implies** A$\cup$B is finite.

Theorem FINSET_1:15. A is finite **implies** A$\cap$B is finite & B$\cap$A is finite.

Theorem FINSET_1:16. A is finite **implies** A$\setminus$B is finite.

Theorem FINSET_1:17. A is finite **implies** f**.**A is finite.

Theorem FINSET_1:18. A is finite **implies** **for** X **being** Subset-Family **of** A **st** X $\neq \emptyset$ **ex** x **being** set **st** x $\in$ X & **for** B **being** set **st** B $\in$ X **holds** x $\subseteq$ B **implies** B = x.

scheme Finite{A() $\rightarrow$ set, P[set]}: P[A()] **provided** A: A() is finite **and** B: P[$\emptyset$] **and** C: **for** x, B **being** set **st** x $\in$ A() & B $\subseteq$ A() & P[B] **holds** P[B$\cup${x}].

Theorem FINSET_1:19. A is finite & B is finite **implies** $[\![$A, B$]\!]$ is finite.

Theorem FINSET_1:20. A is finite & B is finite & C is finite **implies** $[\![$A, B, C$]\!]$ is finite.

Theorem FINSET_1:21. A is finite & B is finite & C is finite & D is finite **implies** $[\![$A, B, C, D$]\!]$ is finite.

Theorem FINSET_1:22. B $\neq \emptyset$ & $[\![$A, B$]\!]$ is finite **implies** A is finite.

Theorem FINSET_1:23. A $\neq \emptyset$ & $[\![$A, B$]\!]$ is finite **implies** B is finite.

Theorem FINSET_1:24. A is finite **iff** bool A is finite.

Theorem FINSET_1:25. A is finite & (**for** X **st** X $\in$ A **holds** X is finite) **iff** $\bigcup$A is finite.

Theorem FINSET_1:26. dom f is finite **implies** rng f is finite.

Theorem FINSET_1:27. Y $\subseteq$ rng f & f^{-1}Y is finite **implies** Y is finite.

Chapter 24

DOMAIN_1

Domains and Their Cartesian Products

by

Andrzej Trybulec [1]

Warsaw University (Białystok)

Summary. The article includes: theorems related to domains, theorems related to Cartesian products presented earlier in various articles and simplified here by substituting domains for sets and omitting the assumption that the sets involved must not be empty. Several schemes and theorems related to Frænkel operator are given. We also redefine subset yielding functions such as the pair of elements of a set and the union of two subsets of a set.

The symbols used in this article are introduced in the following vocabularies: BOOLE, COORD, and SUB_OP. The terminology and notation used in this article have been introduced in the following articles: TARSKI, BOOLE, ENUMSET1, SUBSET_1, FUNCT_1, ORDINAL1, and MCART_1.

reserve a, b, c, d **for** Any, A, B, C **for** set.

reserve D, X1, X2, X3, X4, Y1, Y2, Y3, Y4 **for** DOMAIN.

reserve x1, y1, z1 **for** (Element **of** X1), x2, y2, z2 **for** (Element **of** X2), x3, y3, z3 **for** (Element **of** X3), x4, y4, z4 **for** (Element **of** X4).

Theorem DOMAIN_1:1. A **is** DOMAIN **iff** $A \neq \emptyset$.

Theorem DOMAIN_1:2. $D \neq \emptyset$.

[1]Supported by RPBP.III-24.C1.

Theorem DOMAIN_1:3. a **is** Element **of** D **implies** $a \in D$.

reserve A1, B1 **for** Subset **of** X1.

Theorem DOMAIN_1:4. $A1 = B1^c$ **iff for** x1 **holds** $x1 \in A1$ **iff not** $x1 \in B1$.

Theorem DOMAIN_1:5. $A1 = B1^c$ **iff for** x1 **holds not** $x1 \in A1$ **iff** $x1 \in B1$.

Theorem DOMAIN_1:6. $A1 = B1^c$ **iff for** x1 **holds not** ($x1 \in A1$ **iff** $x1 \in B1$).

Theorem DOMAIN_1:7. $[x1, x2] \in [\![X1, X2]\!]$.

Theorem DOMAIN_1:8. [x1, x2] **is** Element **of** $[\![X1, X2]\!]$.

Theorem DOMAIN_1:9. $a \in [\![X1, X2]\!]$ **implies ex** x1, x2 **st** $a = [x1, x2]$.

reserve x **for** Element **of** $[\![X1, X2]\!]$.

Theorem DOMAIN_1:10. $x = [x_1, x_2]$.

Theorem DOMAIN_1:11. $x \neq x_1$ & $x \neq x_2$.

Theorem DOMAIN_1:12. **for** x, y **being** Element **of** $[\![X1, X2]\!]$ **st** $x_1 = y_1$ & $x_2 = y_2$ **holds** $x = y$.

Theorem DOMAIN_1:13. $[\![A, D]\!] \subseteq [\![B, D]\!]$ **or** $[\![D, A]\!] \subseteq [\![D, B]\!]$ **implies** $A \subseteq B$.

Theorem DOMAIN_1:14. $[\![X1, X2]\!] = [\![A, B]\!]$ **implies** $X1 = A$ & $X2 = B$.

Definition

let X1, X2, x1, x2.

redefine

func [x1, x2] → Element **of** $[\![X1, X2]\!]$.

Definition

let X1, X2.

let x **be** Element **of** $[\![X1, X2]\!]$.

redefine

func x_1 → Element **of** X1.

func x_2 → Element **of** X2.

Theorem DOMAIN_1:15. $a \in [\![X1, X2, X3]\!]$ **iff ex** x1, x2, x3 **st** $a = [x1, x2, x3]$.

Theorem DOMAIN_1:16. (**for** a **holds** $a \in D$ **iff ex** x1, x2, x3 **st** $a = [x1, x2, x3]$) **implies** $D = [\![X1, X2, X3]\!]$.

Theorem DOMAIN_1:17. $D = [\![X1, X2, X3]\!]$ **iff for** a **holds** $a \in D$ **iff ex** x1, x2, x3 **st** $a = [x1, x2, x3]$.

Theorem DOMAIN_1:18. $[\![X1, X2, X3]\!] = [\![Y1, Y2, Y3]\!]$ **implies** $X1 = Y1$ & $X2 = Y2$ & $X3 = Y3$.

reserve x, y **for** Element **of** $[\![X1, X2, X3]\!]$.

Theorem DOMAIN_1:19. $x = [a, b, c]$ **implies** $x_1 = a$ & $x_2 = b$ & $x_3 = c$.

Theorem DOMAIN_1:20. $x = [x_1, x_2, x_3]$.

Theorem DOMAIN_1:21. x_1 = (x **qua** Any)$_{11}$ & x_2 = (x **qua** Any)$_{12}$ & x_3 = (x **qua** Any)$_2$.

Theorem DOMAIN_1:22. x ≠ x_1 & x ≠ x_2 & x ≠ x_3.

Theorem DOMAIN_1:23. [x1, x2, x3] ∈ ⟦X1, X2, X3⟧.

Definition
let X1, X2, X3, x1, x2, x3.
redefine
func [x1, x2, x3] → Element **of** ⟦X1, X2, X3⟧.

Definition
let X1, X2, X3.
let x **be** Element **of** ⟦X1, X2, X3⟧.
redefine
func x_1 → Element **of** X1.
func x_2 → Element **of** X2.
func x_3 → Element **of** X3.

Theorem DOMAIN_1:24. a = x_1 **iff for** x1, x2, x3 **st** x = [x1, x2, x3] **holds** a = x1.

Theorem DOMAIN_1:25. b = x_2 **iff for** x1, x2, x3 **st** x = [x1, x2, x3] **holds** b = x2.

Theorem DOMAIN_1:26. c = x_3 **iff for** x1, x2, x3 **st** x = [x1, x2, x3] **holds** c = x3.

Theorem DOMAIN_1:27. [x_1, x_2, x_3] = x.

Theorem DOMAIN_1:28. x_1 = y_1 & x_2 = y_2 & x_3 = y_3 **implies** x = y.

Theorem DOMAIN_1:29. [x1, x2, x3]$_1$ = x1 & [x1, x2, x3]$_2$ = x2 & [x1, x2, x3]$_3$ = x3.

Theorem DOMAIN_1:30. **for** x **being** (Element **of** ⟦X1, X2, X3⟧), y **being** Element **of** ⟦Y1, Y2, Y3⟧ **holds** x = y **implies** x_1 = y_1 & x_2 = y_2 & x_3 = y_3.

Theorem DOMAIN_1:31. a ∈ ⟦X1, X2, X3, X4⟧ **iff ex** x1, x2, x3, x4 **st** a = [x1, x2, x3, x4].

Theorem DOMAIN_1:32. (**for** a **holds** a ∈ D **iff ex** x1, x2, x3, x4 **st** a = [x1, x2, x3, x4]) **implies** D = ⟦X1, X2, X3, X4⟧.

Theorem DOMAIN_1:33. D = ⟦X1, X2, X3, X4⟧ **iff for** a **holds** a ∈ D **iff ex** x1, x2, x3, x4 **st** a = [x1, x2, x3, x4].

reserve x, y **for** Element **of** ⟦X1, X2, X3, X4⟧.

Theorem DOMAIN_1:34. ⟦X1, X2, X3, X4⟧ = ⟦Y1, Y2, Y3, Y4⟧ **implies** X1 = Y1 & X2 = Y2 & X3 = Y3 & X4 = Y4.

Theorem DOMAIN_1:35. x = [a, b, c, d] **implies** x_1 = a & x_2 = b & x_3 = c & x_4 = d.

Theorem DOMAIN_1:36. x = [x_1, x_2, x_3, x_4].

Theorem DOMAIN_1:37. x_1 = (x **qua** Any)$_{111}$ & x_2 = (x **qua** Any)$_{112}$ & x_3 = (x **qua** Any)$_{12}$ & x_4 = (x **qua** Any)$_2$.

Theorem DOMAIN_1:38. $x \neq x_1$ & $x \neq x_2$ & $x \neq x_3$ & $x \neq x_4$.

Theorem DOMAIN_1:39. [x1, x2, x3, x4] ∈ ⟦X1, X2, X3, X4⟧.

Definition

let X1, X2, X3, X4, x1, x2, x3, x4.

redefine

func [x1, x2, x3, x4] → Element **of** ⟦X1, X2, X3, X4⟧.

Definition

let X1, X2, X3, X4.

let x **be** Element **of** ⟦X1, X2, X3, X4⟧.

redefine

func x_1 → Element **of** X1.

func x_2 → Element **of** X2.

func x_3 → Element **of** X3.

func x_4 → Element **of** X4.

Theorem DOMAIN_1:40. a = x_1 **iff for** x1, x2, x3, x4 **st** x = [x1, x2, x3, x4] **holds** a = x1.

Theorem DOMAIN_1:41. b = x_2 **iff for** x1, x2, x3, x4 **st** x = [x1, x2, x3, x4] **holds** b = x2.

Theorem DOMAIN_1:42. c = x_3 **iff for** x1, x2, x3, x4 **st** x = [x1, x2, x3, x4] **holds** c = x3.

Theorem DOMAIN_1:43. d = x_4 **iff for** x1, x2, x3, x4 **st** x = [x1, x2, x3, x4] **holds** d = x4.

Theorem DOMAIN_1:44. **for** x **being** Element **of** ⟦X1, X2, X3, X4⟧ **holds** [x_1, x_2, x_3, x_4] = x.

Theorem DOMAIN_1:45. **for** x, y **being** Element **of** ⟦X1, X2, X3, X4⟧ **st** $x_1 = y_1$ & $x_2 = y_2$ & $x_3 = y_3$ & $x_4 = y_4$ **holds** x = y.

Theorem DOMAIN_1:46. $[x1, x2, x3, x4]_1$ = x1 & $[x1, x2, x3, x4]_2$ = x2 & $[x1, x2, x3, x4]_3$ = x3 & $[x1, x2, x3, x4]_4$ = x4.

Theorem DOMAIN_1:47. **for** x **being** (Element **of** ⟦X1, X2, X3, X4⟧), y **being** Element **of** ⟦Y1, Y2, Y3, Y4⟧ **holds** x = y **implies** $x_1 = y_1$ & $x_2 = y_2$ & $x_3 = y_3$ & $x_4 = y_4$.

reserve A2 **for** (Subset **of** X2), A3 **for** (Subset **of** X3), A4 **for** Subset **of** X4.

scheme Fraenkel1{P[Any]}: **for** X1 **holds** {x1: P[x1]} **is** Subset **of** X1.

scheme Fraenkel2{P[Any, Any]}: **for** X1, X2 **holds** {[x1, x2]: P[x1, x2]} **is** Subset **of** ⟦X1, X2⟧.

scheme Fraenkel3{P[Any, Any, Any]}: **for** X1, X2, X3 **holds** {[x1, x2, x3]: P[x1, x2, x3]} **is** Subset **of** ⟦X1, X2, X3⟧.

scheme Fraenkel4{P[Any, Any, Any, Any]}: **for** X1, X2, X3, X4 **holds** {[x1, x2, x3, x4]: P[x1, x2, x3, x4]} **is** Subset **of** ⟦X1, X2, X3, X4⟧.

scheme Fraenkel5{P[Any], Q[Any]}: **for** X1 **st** **for** x1 **holds** P[x1] **implies** Q[x1] **holds** {y1: P[y1]} ⊆ {z1: Q[z1]}.

scheme Fraenkel6{P[Any], Q[Any]}: **for** X1 **st** **for** x1 **holds** P[x1] **iff** Q[x1] **holds** {y1: P[y1]} = {z1: Q[z1]}.

Theorem DOMAIN_1:48. X1 = {x1: **not contradiction**}.

Theorem DOMAIN_1:49. ⟦X1, X2⟧ = {[x1, x2]: **not contradiction**}.

Theorem DOMAIN_1:50. ⟦X1, X2, X3⟧ = {[x1, x2, x3]: **not contradiction**}.

Theorem DOMAIN_1:51. ⟦X1, X2, X3, X4⟧ = {[x1, x2, x3, x4]: **not contradiction**}.

Theorem DOMAIN_1:52. A1 = {x1: x1 ∈ A1}.

Definition

let X1, X2, A1, A2.

redefine

func ⟦A1, A2⟧ → Subset **of** ⟦X1, X2⟧.

Theorem DOMAIN_1:53. ⟦A1, A2⟧ = {[x1, x2]: x1 ∈ A1 & x2 ∈ A2}.

Definition

let X1, X2, X3, A1, A2, A3.

redefine

func ⟦A1, A2, A3⟧ → Subset **of** ⟦X1, X2, X3⟧.

Theorem DOMAIN_1:54. ⟦A1, A2, A3⟧ = {[x1, x2, x3]: x1 ∈ A1 & x2 ∈ A2 & x3 ∈ A3}.

Definition

let X1, X2, X3, X4, A1, A2, A3, A4.

redefine

func ⟦A1, A2, A3, A4⟧ → Subset **of** ⟦X1, X2, X3, X4⟧.

Theorem DOMAIN_1:55. ⟦A1, A2, A3, A4⟧ = {[x1, x2, x3, x4]: x1 ∈ A1 & x2 ∈ A2 & x3 ∈ A3 & x4 ∈ A4}.

Theorem DOMAIN_1:56. ∅ X1 = {x1: **contradiction**}.

Theorem DOMAIN_1:57. $A1^c$ = {x1: **not** x1 ∈ A1}.

Theorem DOMAIN_1:58. A1∩B1 = {x1: x1 ∈ A1 & x1 ∈ B1}.

Theorem DOMAIN_1:59. A1∪B1 = {x1: x1 ∈ A1 **or** x1 ∈ B1}.

Theorem DOMAIN_1:60. A1∖B1 = {x1: x1 ∈ A1 & **not** x1 ∈ B1}.

Theorem DOMAIN_1:61. A1∸B1 = {x1: x1 ∈ A1 & **not** x1 ∈ B1 **or** **not** x1 ∈ A1 & x1 ∈ B1}.

Theorem DOMAIN_1:62. A1∸B1 = {x1: **not** x1 ∈ A1 **iff** x1 ∈ B1}.

Theorem DOMAIN_1:63. A1$\dot{-}$B1 = {x1: x1 ∈ A1 **iff** **not** x1 ∈ B1}.

Theorem DOMAIN_1:64. A1$\dot{-}$B1 = {x1: **not** (x1 ∈ A1 **iff** x1 ∈ B1)}.

reserve x1, x2, x3, x4, x5, x6, x7, x8 **for** Element **of** D.

Theorem DOMAIN_1:65. {x1} **is** Subset **of** D.

Theorem DOMAIN_1:66. {x1, x2} **is** Subset **of** D.

Theorem DOMAIN_1:67. {x1, x2, x3} **is** Subset **of** D.

Theorem DOMAIN_1:68. {x1, x2, x3, x4} **is** Subset **of** D.

Theorem DOMAIN_1:69. {x1, x2, x3, x4, x5} **is** Subset **of** D.

Theorem DOMAIN_1:70. {x1, x2, x3, x4, x5, x6} **is** Subset **of** D.

Theorem DOMAIN_1:71. {x1, x2, x3, x4, x5, x6, x7} **is** Subset **of** D.

Theorem DOMAIN_1:72. {x1, x2, x3, x4, x5, x6, x7, x8} **is** Subset **of** D.

Definition

let D.

redefine

let x1 **be** Element **of** D.

func {x1} → Subset **of** D.

let x2 **be** Element **of** D.

func {x1, x2} → Subset **of** D.

let x3 **be** Element **of** D.

func {x1, x2, x3} → Subset **of** D.

let x4 **be** Element **of** D.

func {x1, x2, x3, x4} → Subset **of** D.

let x5 **be** Element **of** D.

func {x1, x2, x3, x4, x5} → Subset **of** D.

let x6 **be** Element **of** D.

func {x1, x2, x3, x4, x5, x6} → Subset **of** D.

let x7 **be** Element **of** D.

func {x1, x2, x3, x4, x5, x6, x7} → Subset **of** D.

let x8 **be** Element **of** D.

func {x1, x2, x3, x4, x5, x6, x7, x8} → Subset **of** D.

Definition

let X1, A1.

redefine

func A1^c → Subset **of** X1.

let B1.

func $A1 \cup B1 \to$ Subset **of** X1.
func $A1 \cap B1 \to$ Subset **of** X1.
func $A1 \setminus B1 \to$ Subset **of** X1.
func $A1 \dot{-} B1 \to$ Subset **of** X1.

Chapter 25

FINSUB_1

Boolean Domains

by

Andrzej Trybulec [1]

Warsaw University (Białystok)

Agata Darmochwał[2]

Warsaw University (Białystok)

Summary. BOOLE DOMAIN is a SET DOMAIN that is closed under union and difference. This condition is equivalent to being closed under symmetric difference and one of the following operations: union, intersection or difference. We introduce the set of all finite subsets of a set A, denoted by Fin A. The mode Finite Subset of a set A is introduced with the mother type: Element of Fin A. In consequence, "Finite Subset of ..." is an elementary type, therefore one may use such types as "set of Finite Subset of A", "[(Finite Subset of A), Finite Subset of A]", and so on. The article begins with some auxiliary theorems that belong really to BOOLE or ORDINAL1 but are missing there. Moreover, bool A is redefined as a SET DOMAIN, for an arbitrary set A.

The symbols used in this article are introduced in the following vocabularies: BOOLE, FINITE, and BOOLEDOM. The terminology and notation used in this article have been introduced in the following articles: TARSKI, BOOLE, FUNCT_1, REAL_1, NAT_1, FINSEQ_1, ENUMSET1, SUBSET_1, ORDINAL1, MCART_1, SETFAM_1, FINSET_1, and DOMAIN_1.

[1] Supported by RPBP.III-24.C1.

[2] Supported by RPBP.III-24.C1.

reserve X, Y **for** set.

Theorem FINSUB_1:1. X misses Y **implies** X\Y = X & Y\X = Y.

Theorem FINSUB_1:2. X misses Y **implies** (X∪Y)\Y = X & (X∪Y)\X = Y.

Theorem FINSUB_1:3. X∪Y = X∸(Y\X).

Theorem FINSUB_1:4. X∪Y = X∸Y∸X∩Y.

Theorem FINSUB_1:5. X\Y = X∸(X∩Y).

Theorem FINSUB_1:6. X∩Y = X∸Y∸(X∪Y).

Theorem FINSUB_1:7. (**for** x **being** set **st** x ∈ X **holds** x ∈ Y) **implies** X ⊆ Y.

Definition

let X.

redefine

func bool X → SET DOMAIN.

Theorem FINSUB_1:8. **for** Y **being** Element **of** bool X **holds** Y ⊆ X.

Definition

mode BOOLE DOMAIN → SET DOMAIN **means for** X, Y **being** Element **of it holds** X∪Y ∈ **it** & X\Y ∈ **it**.

Theorem FINSUB_1:9. **for** A **being** SET DOMAIN **holds** A **is** BOOLE DOMAIN **iff for** X, Y **being** Element **of** A **holds** X∪Y ∈ A & X\Y ∈ A.

reserve A **for** BOOLE DOMAIN.

Theorem FINSUB_1:10. X ∈ A & Y ∈ A **implies** X∪Y ∈ A & X\Y ∈ A.

Theorem FINSUB_1:11. X **is** Element **of** A & Y **is** Element **of** A **implies** X∪Y **is** Element **of** A.

Theorem FINSUB_1:12. X **is** Element **of** A & Y **is** Element **of** A **implies** X\Y **is** Element **of** A.

Definition

let A.

let X, Y **be** Element **of** A.

redefine

func X∪Y → Element **of** A.

func X\Y → Element **of** A.

Theorem FINSUB_1:13. X **is** Element **of** A & Y **is** Element **of** A **implies** X∩Y **is** Element **of** A.

Theorem FINSUB_1:14. X **is** Element **of** A & Y **is** Element **of** A **implies** X∸Y **is** Element **of** A.

Theorem FINSUB_1:15. **for** A **being** SET DOMAIN **st for** X, Y **being** Element **of** A **holds** X∸Y ∈ A & X\Y ∈ A **holds** A **is** BOOLE DOMAIN.

Theorem FINSUB_1:16. **for** A **being** SET DOMAIN **st for** X, Y **being** Element **of** A **holds** X∸Y ∈ A & X∩Y ∈ A **holds** A **is** BOOLE DOMAIN.

Theorem FINSUB_1:17. **for** A **being** SET DOMAIN **st for** X, Y **being** Element **of** A **holds** X∸Y ∈ A & X∪Y ∈ A **holds** A **is** BOOLE DOMAIN.

Definition

let A.

let X, Y **be** Element **of** A.

redefine

func X∩Y → Element **of** A.

func X∸Y → Element **of** A.

Theorem FINSUB_1:18. ∅ ∈ A.

Theorem FINSUB_1:19. ∅ **is** Element **of** A.

Theorem FINSUB_1:20. bool A **is** BOOLE DOMAIN.

Theorem FINSUB_1:21. **for** A, B **being** BOOLE DOMAIN **holds** A∩B **is** BOOLE DOMAIN.

reserve A, B, P **for** set.

reserve x, y **for** Any.

Definition

let A.

func Fin A → BOOLE DOMAIN **means for** X **being** set **holds** X ∈ **it iff** X ⊆ A & X is finite.

Theorem FINSUB_1:22. B ∈ Fin A **iff** B ⊆ A & B is finite.

Theorem FINSUB_1:23. A ⊆ B **implies** Fin A ⊆ Fin B.

Theorem FINSUB_1:24. Fin (A∩B) = Fin A∩Fin B.

Theorem FINSUB_1:25. Fin A∪Fin B ⊆ Fin (A∪B).

Theorem FINSUB_1:26. Fin A ⊆ bool A.

Theorem FINSUB_1:27. A is finite **implies** Fin A = bool A.

Theorem FINSUB_1:28. Fin ∅ = {∅}.

Definition

let A.

mode Finite Subset **of** A → Element **of** Fin A **means not contradiction**.

Theorem FINSUB_1:29. **for** X **being** Element **of** Fin A **holds** X **is** Finite Subset **of** A.

Definition

let A.

let X, Y **be** Finite Subset **of** A.

redefine

func X∪Y → Finite Subset **of** A.

func X∩Y → Finite Subset **of** A.

func X \ Y → Finite Subset **of** A.

func X∸Y → Finite Subset **of** A.

Theorem FINSUB_1:30. **for** X **being** Finite Subset **of** A **holds** X is finite.

Theorem FINSUB_1:31. **for** X **being** Finite Subset **of** A **holds** X ⊆ A.

Theorem FINSUB_1:32. **for** X **being** Finite Subset **of** A **holds** X **is** Subset **of** A.

Theorem FINSUB_1:33. ∅ **is** Finite Subset **of** A.

Theorem FINSUB_1:34. A is finite **implies for** X **being** Subset **of** A **holds** X **is** Finite Subset **of** A.

Chapter 26

INCSP_1

Axioms of Incidency

by

Wojciech A. Trybulec [1]

Warsaw University

Summary. This text is a translation into Mizar of a small part of *Foundations of Geometry* by K. Borsuk and W. Szmielew related to the axioms of incidency. (Remark: The fourth axiom of incidency is weakened in this text. In the source text it has the form: *for any plane there exist three non-collinear points in the plane* and in this text: *for any plane there exists one point in the plane.* The original axiom is proved in the text.) The article includes: theorems concerning collinearity of points and coplanarity of points and lines, basic theorems concerning lines and planes, fundamental existence theorems, theorems concerning intersection of lines and planes.

The symbols used in this article are introduced in the following vocabularies: INCSP_1, BOOLE, and RELATION. The terminology and notation used in this article have been introduced in the following articles: TARSKI, BOOLE, ENUMSET1, SUBSET_1, RELAT_1, MCART_1, DOMAIN_1, and RELSET_1.

struct IncStruct ⟨⟨Points, Lines, Planes → DOMAIN, Inc1 → (Relation **of the** Points, **the** Lines), Inc2 → (Relation **of the** Points, **the** Planes), Inc3 → Relation **of the** Lines, **the** Planes⟩⟩.

[1] Supported by RPBP.III-24.C1.

Definition

let S **be** IncStruct.

mode POINT **of** S → Element **of the** Points **of** S **means not contradiction**.

mode LINE **of** S → Element **of the** Lines **of** S **means not contradiction**.

mode PLANE **of** S → Element **of the** Planes **of** S **means not contradiction**.

reserve S **for** IncStruct.

reserve A **for** Element **of the** Points **of** S.

reserve L **for** Element **of the** Lines **of** S.

reserve P **for** Element **of the** Planes **of** S.

Theorem INCSP_1:1. A **is** POINT **of** S.

Theorem INCSP_1:2. L **is** LINE **of** S.

Theorem INCSP_1:3. P **is** PLANE **of** S.

reserve A, B, C, D, E **for** POINT **of** S.

reserve K, L, L1, L2 **for** LINE **of** S.

reserve P, P1, P2, Q **for** PLANE **of** S.

reserve F, G **for** Subset **of the** Points **of** S.

Definition

let S.

let A **be** (POINT **of** S), L **be** LINE **of** S.

pred A on L **means** [A, L] ∈ **the** Inc1 **of** S.

Definition

let S.

let A **be** (POINT **of** S), P **be** PLANE **of** S.

pred A on P **means** [A, P] ∈ **the** Inc2 **of** S.

Definition

let S.

let L **be** (LINE **of** S), P **be** PLANE **of** S.

pred L on P **means** [L, P] ∈ **the** Inc3 **of** S.

Definition

let S.

let F **be** (**set of** POINT **of** S), L **be** LINE **of** S.

pred F on L **means for** A **being** POINT **of** S **st** A ∈ F **holds** A on L.

Definition

let S.

let F **be** (**set of** POINT **of** S), P **be** PLANE **of** S.

pred F on P **means for** A **st** A ∈ F **holds** A on P.

Definition

let S.

let F **be set of** POINT **of** S.

pred F is collinear **means ex** L **st** F on L.

Definition

let S.

let F **be set of** POINT **of** S.

pred F is coplanar **means ex** P **st** F on P.

Theorem INCSP_1:4. A on L **iff** [A, L] ∈ **the** Inc1 **of** S.

Theorem INCSP_1:5. A on P **iff** [A, P] ∈ **the** Inc2 **of** S.

Theorem INCSP_1:6. L on P **iff** [L, P] ∈ **the** Inc3 **of** S.

Theorem INCSP_1:7. F on L **iff for** A **st** A ∈ F **holds** A on L.

Theorem INCSP_1:8. F on P **iff for** A **st** A ∈ F **holds** A on P.

Theorem INCSP_1:9. F is collinear **iff ex** L **st** F on L.

Theorem INCSP_1:10. F is coplanar **iff ex** P **st** F on P.

Theorem INCSP_1:11. {A, B} on L **iff** A on L & B on L.

Theorem INCSP_1:12. {A, B, C} on L **iff** A on L & B on L & C on L.

Theorem INCSP_1:13. {A, B} on P **iff** A on P & B on P.

Theorem INCSP_1:14. {A, B, C} on P **iff** A on P & B on P & C on P.

Theorem INCSP_1:15. {A, B, C, D} on P **iff** A on P & B on P & C on P & D on P.

Theorem INCSP_1:16. G ⊆ F & F on L **implies** G on L.

Theorem INCSP_1:17. G ⊆ F & F on P **implies** G on P.

Theorem INCSP_1:18. F on L & A on L **iff** F∪{A} on L.

Theorem INCSP_1:19. F on P & A on P **iff** F∪{A} on P.

Theorem INCSP_1:20. F∪G on L **iff** F on L & G on L.

Theorem INCSP_1:21. F∪G on P **iff** F on P & G on P.

Theorem INCSP_1:22. G ⊆ F & F is collinear **implies** G is collinear.

Theorem INCSP_1:23. G ⊆ F & F is coplanar **implies** G is coplanar.

Definition

mode IncSpace → IncStruct **means** (**for** L **being** LINE **of it ex** A, B **being** POINT **of it st** A ≠ B & {A, B} on L) & (**for** A, B **being** POINT **of it ex** L **being** LINE **of it st** {A, B} on L) & (**for** A, B **being** (POINT **of it**), K, L **being** LINE **of it st** A ≠ B & {A, B} on K & {A, B} on L **holds** K = L) & (**for** P **being** PLANE **of it ex** A **being** POINT **of it st** A on P) & (**for** A, B, C **being** POINT **of it ex** P **being** PLANE **of it st** {A, B, C} on P) & (**for** A, B, C **being** (POINT **of it**), P, Q **being** PLANE **of it st not**

{A, B, C} is collinear & {A, B, C} on P & {A, B, C} on Q **holds** P = Q) & (**for** L **being** (LINE **of it**), P **being** PLANE **of it st ex** A, B **being** POINT **of it st** A ≠ B & {A, B} on L & {A, B} on P **holds** L on P) & (**for** A **being** (POINT **of it**), P, Q **being** PLANE **of it st** A on P & A on Q **ex** B **being** POINT **of it st** A ≠ B & B on P & B on Q) & (**ex** A, B, C, D **being** POINT **of it st not** {A, B, C, D} is coplanar) & (**for** A **being** (POINT **of it**), L **being** (LINE **of it**), P **being** PLANE **of it st** A on L & L on P **holds** A on P).

Theorem INCSP_1:24. (**for** L **being** LINE **of** S **ex** A, B **being** POINT **of** S **st** A ≠ B & {A, B} on L) & (**for** A, B **being** POINT **of** S **ex** L **being** LINE **of** S **st** {A, B} on L) & (**for** A, B **being** (POINT **of** S), K, L **being** LINE **of** S **st** A ≠ B & {A, B} on K & {A, B} on L **holds** K = L) & (**for** P **being** PLANE **of** S **ex** A **being** POINT **of** S **st** A on P) & (**for** A, B, C **being** POINT **of** S **ex** P **being** PLANE **of** S **st** {A, B, C} on P) & (**for** A, B, C **being** (POINT **of** S), P, Q **being** PLANE **of** S **st not** {A, B, C} is collinear & {A, B, C} on P & {A, B, C} on Q **holds** P = Q) & (**for** L **being** (LINE **of** S), P **being** PLANE **of** S **st ex** A, B **being** POINT **of** S **st** A ≠ B & {A, B} on L & {A, B} on P **holds** L on P) & (**for** A **being** (POINT **of** S), P, Q **being** PLANE **of** S **st** A on P & A on Q **ex** B **being** POINT **of** S **st** A ≠ B & B on P & B on Q) & (**ex** A, B, C, D **being** POINT **of** S **st not** {A, B, C, D} is coplanar) & (**for** A **being** (POINT **of** S), L **being** (LINE **of** S), P **being** PLANE **of** S **st** A on L & L on P **holds** A on P) **implies** S **is** IncSpace.

reserve S **for** IncSpace.

reserve A, B, C, D, E **for** POINT **of** S.

reserve K, L, L1, L2 **for** LINE **of** S.

reserve P, P1, P2, Q **for** PLANE **of** S.

reserve F **for** Subset **of the** Points **of** S.

Theorem INCSP_1:25. **ex** A, B **st** A ≠ B & {A, B} on L.

Theorem INCSP_1:26. **ex** L **st** {A, B} on L.

Theorem INCSP_1:27. A ≠ B & {A, B} on K & {A, B} on L **implies** K = L.

Theorem INCSP_1:28. **ex** A **st** A on P.

Theorem INCSP_1:29. **ex** P **st** {A, B, C} on P.

Theorem INCSP_1:30. **not** {A, B, C} is collinear & {A, B, C} on P & {A, B, C} on Q **implies** P = Q.

Theorem INCSP_1:31. (**ex** A, B **st** A ≠ B & {A, B} on L & {A, B} on P) **implies** L on P.

Theorem INCSP_1:32. A on P & A on Q **implies** (**ex** B **st** A ≠ B & B on P & B on Q).

Theorem INCSP_1:33. **ex** A, B, C, D **st not** {A, B, C, D} is coplanar.

Theorem INCSP_1:34. A on L & L on P **implies** A on P.

Theorem INCSP_1:35. F on L & L on P **implies** F on P.

Theorem INCSP_1:36. {A, A, B} is collinear.

Theorem INCSP_1:37. {A, A, B, C} is coplanar.

Theorem INCSP_1:38. {A, B, C} is collinear **implies** {A, B, C, D} is coplanar.

Theorem INCSP_1:39. A ≠ B & {A, B} on L & **not** C on L **implies not** {A, B, C} is collinear.

Theorem INCSP_1:40. **not** {A, B, C} is collinear & {A, B, C} on P & **not** D on P **implies not** {A, B, C, D} is coplanar.

Theorem INCSP_1:41. **not** (**ex** P **st** K on P & L on P) **implies** K ≠ L.

Theorem INCSP_1:42. **not** (**ex** P **st** L on P & L1 on P & L2 on P) & (**ex** A **st** A on L & A on L1 & A on L2) **implies** L ≠ L1.

Theorem INCSP_1:43. L1 on P & L2 on P & **not** L on P & L1 ≠ L2 **implies not** (**ex** Q **st** L on Q & L1 on Q & L2 on Q).

Theorem INCSP_1:44. **ex** P **st** A on P & L on P.

Theorem INCSP_1:45. (**ex** A **st** A on K & A on L) **implies** (**ex** P **st** K on P & L on P).

Theorem INCSP_1:46. A ≠ B **implies ex** L **st for** K **holds** {A, B} on K **iff** K = L.

Theorem INCSP_1:47. **not** {A, B, C} is collinear **implies ex** P **st for** Q **holds** {A, B, C} on Q **iff** P = Q.

Theorem INCSP_1:48. **not** A on L **implies ex** P **st for** Q **holds** A on Q & L on Q **iff** P = Q.

Theorem INCSP_1:49. K ≠ L & (**ex** A **st** A on K & A on L) **implies ex** P **st for** Q **holds** K on Q & L on Q **iff** P = Q.

Definition

let S.

let A, B.

assume A ≠ B.

func Line (A, B) → LINE **of** S **means** {A, B} on **it**.

Definition

let S.

let A, B, C.

assume not {A, B, C} is collinear.

func Plane (A, B, C) → PLANE **of** S **means** {A, B, C} on **it**.

Definition

let S.

let A, L.

assume not A on L.

func Plane (A, L) → PLANE **of** S **means** A on **it** & L on **it**.

Definition

let S.

let K, L.

assume that K $\neq$ L **and** (**ex** A **st** A on K & A on L).

func Plane (K, L) $\rightarrow$ PLANE **of** S **means** K on **it** & L on **it**.

Theorem INCSP_1:50. A $\neq$ B **implies** {A, B} on Line (A, B).

Theorem INCSP_1:51. A $\neq$ B & {A, B} on K **implies** K = Line (A, B).

Theorem INCSP_1:52. **not** {A, B, C} is collinear **implies** {A, B, C} on Plane (A, B, C).

Theorem INCSP_1:53. **not** {A, B, C} is collinear & {A, B, C} on Q **implies** Q = Plane (A, B, C).

Theorem INCSP_1:54. **not** A on L **implies** A on Plane (A, L) & L on Plane (A, L).

Theorem INCSP_1:55. **not** A on L & A on Q & L on Q **implies** Q = Plane (A, L).

Theorem INCSP_1:56. K $\neq$ L & (**ex** A **st** A on K & A on L) **implies** K on Plane (K, L) & L on Plane (K, L).

Theorem INCSP_1:57. A $\neq$ B **implies** Line (A, B) = Line (B, A).

Theorem INCSP_1:58. **not** {A, B, C} is collinear **implies** Plane (A, B, C) = Plane (A, C, B).

Theorem INCSP_1:59. **not** {A, B, C} is collinear **implies** Plane (A, B, C) = Plane (B, A, C).

Theorem INCSP_1:60. **not** {A, B, C} is collinear **implies** Plane (A, B, C) = Plane (B, C, A).

Theorem INCSP_1:61. **not** {A, B, C} is collinear **implies** Plane (A, B, C) = Plane (C, A, B).

Theorem INCSP_1:62. **not** {A, B, C} is collinear **implies** Plane (A, B, C) = Plane (C, B, A).

Theorem INCSP_1:63. K $\neq$ L & (**ex** A **st** A on K & A on L) & K on Q & L on Q **implies** Q = Plane (K, L).

Theorem INCSP_1:64. K $\neq$ L & (**ex** A **st** A on K & A on L) **implies** Plane (K, L) = Plane (L, K).

Theorem INCSP_1:65. A $\neq$ B & C on Line (A, B) **implies** {A, B, C} is collinear.

Theorem INCSP_1:66. A $\neq$ B & A $\neq$ C & {A, B, C} is collinear **implies** Line (A, B) = Line (A, C).

Theorem INCSP_1:67. **not** {A, B, C} is collinear **implies** Plane (A, B, C) = Plane (C, Line (A, B)).

Theorem INCSP_1:68. **not** {A, B, C} is collinear & D on Plane (A, B, C) **implies** {A, B, C, D} is coplanar.

Theorem INCSP_1:69. **not** C on L & {A, B} on L & A ≠ B **implies** Plane (C, L) = Plane (A, B, C).

Theorem INCSP_1:70. **not** {A, B, C} is collinear **implies** Plane (A, B, C) = Plane (Line (A, B), Line (A, C)).

Theorem INCSP_1:71. **ex** A, B, C **st** {A, B, C} on P & **not** {A, B, C} is collinear.

Theorem INCSP_1:72. **ex** A, B, C, D **st** A on P & **not** {A, B, C, D} is coplanar.

Theorem INCSP_1:73. **ex** B **st** A ≠ B & B on L.

Theorem INCSP_1:74. A ≠ B **implies ex** C **st** C on P & **not** {A, B, C} is collinear.

Theorem INCSP_1:75. **not** {A, B, C} is collinear **implies ex** D **st not** {A, B, C, D} is coplanar.

Theorem INCSP_1:76. **ex** B, C **st** {B, C} on P & **not** {A, B, C} is collinear.

Theorem INCSP_1:77. A ≠ B **implies** (**ex** C, D **st not** {A, B, C, D} is coplanar).

Theorem INCSP_1:78. **ex** B, C, D **st not** {A, B, C, D} is coplanar.

Theorem INCSP_1:79. **ex** L **st not** A on L & L on P.

Theorem INCSP_1:80. A on P **implies** (**ex** L, L1, L2 **st** L1 ≠ L2 & L1 on P & L2 on P & **not** L on P & A on L & A on L1 & A on L2).

Theorem INCSP_1:81. **ex** L, L1, L2 **st** A on L & A on L1 & A on L2 & **not** (**ex** P **st** L on P & L1 on P & L2 on P).

Theorem INCSP_1:82. **ex** P **st** A on P & **not** L on P.

Theorem INCSP_1:83. **ex** A **st** A on P & **not** A on L.

Theorem INCSP_1:84. **ex** K **st not** (**ex** P **st** L on P & K on P).

Theorem INCSP_1:85. **ex** P, Q **st** P ≠ Q & L on P & L on Q.

Theorem INCSP_1:86. K ≠ L & {A, B} on K & {A, B} on L **implies** A = B.

Theorem INCSP_1:87. **not** L on P & {A, B} on L & {A, B} on P **implies** A = B.

Theorem INCSP_1:88. P ≠ Q **implies not** (**ex** A **st** A on P & A on Q) **or** (**ex** L **st for** B **holds** B on P & B on Q **iff** B on L).

Chapter 27

LATTICES

Introduction to Lattice Theory

by

Stanisław Żukowski [1]

Warsaw University (Białystok)

Summary. A lattice is defined as an algebra on a nonempty set with binary operations join and meet which are commutative and associative, and satisfy the absorption identities. The following kinds of lattices are considered: distributive, modular, bounded (with zero and unit elements), complemented, and Boolean (with complement). The article includes also theorems which immediately follow from definitions.

The symbols used in this article are introduced in the following vocabularies: BOOLE, COORD, FUNC, SUB_OP, BINOP, FUNC_REL, BOOLEDOM, and LATTICES. The terminology and notation used in this article have been introduced in the following articles: TARSKI, BOOLE, ENUMSET1, SUBSET_1, FUNCT_1, MCART_1, DOMAIN_1, FUNCT_2, BINOP_1, FINSET_1, and FINSUB_1.

scheme BooleDomBinOpLambda{A() → BOOLE DOMAIN, O((Element **of** A()), Element **of** A()) → Element **of** A()}: **ex** o **being** BinOp **of** A() **st for** a, b **being** Element **of** A() **holds** o.(a, b) = O(a, b).

struct LattStr ⟨⟨L carrier → DOMAIN, L join, L meet → BinOp **of the** L carrier⟩⟩.

reserve G **for** LattStr.

[1]Supported by RPBP.III-24.C1.

reserve p, q, r **for** Element **of the** L carrier **of** G.

Definition

let G, p, q.

func p⊔q → Element **of the** L carrier **of** G **means it** = (**the** L join **of** G).(p, q).

func p⊓q → Element **of the** L carrier **of** G **means it** = (**the** L meet **of** G).(p, q).

Theorem LATTICES:1. p⊔q = (**the** L join **of** G).(p, q).

Theorem LATTICES:2. p⊓q = (**the** L meet **of** G).(p, q).

Definition

let G, p, q.

pred p ⊑ q **means** p⊔q = q.

Theorem LATTICES:3. p ⊑ q **iff** p⊔q = q.

Definition

mode Lattice → LattStr **means** (**for** a, b **being** Element **of the** L carrier **of it holds** a⊔b = b⊔a) & (**for** a, b, c **being** Element **of the** L carrier **of it holds** a⊔(b⊔c) = (a⊔b)⊔c) & (**for** a, b **being** Element **of the** L carrier **of it holds** (a⊓b)⊔b = b) & (**for** a, b **being** Element **of the** L carrier **of it holds** a⊓b = b⊓a) & (**for** a, b, c **being** Element **of the** L carrier **of it holds** a⊓(b⊓c) = (a⊓b)⊓c) & (**for** a, b **being** Element **of the** L carrier **of it holds** a⊓(a⊔b) = a).

Theorem LATTICES:4. (**for** p, q **holds** p⊔q = q⊔p) & (**for** p, q, r **holds** p⊔(q⊔r) = (p⊔q)⊔r) & (**for** p, q **holds** (p⊓q)⊔q = q) & (**for** p, q **holds** p⊓q = q⊓p) & (**for** p, q, r **holds** p⊓(q⊓r) = (p⊓q)⊓r) & (**for** p, q **holds** p⊓(p⊔q) = p) **implies** G **is** Lattice.

reserve L **for** Lattice.

reserve a, b, c, c1, c2 **for** Element **of the** L carrier **of** L.

Theorem LATTICES:5. a⊔b = b⊔a.

Theorem LATTICES:6. a⊓b = b⊓a.

Theorem LATTICES:7. a⊔(b⊔c) = (a⊔b)⊔c.

Theorem LATTICES:8. a⊓(b⊓c) = (a⊓b)⊓c.

Theorem LATTICES:9. (a⊓b)⊔b = b & b⊔(a⊓b) = b & b⊔(b⊓a) = b & (b⊓a)⊔b = b.

Theorem LATTICES:10. a⊓(a⊔b) = a & (a⊔b)⊓a = a & (b⊔a)⊓a = a & a⊓(b⊔a) = a.

Definition

mode D Lattice → Lattice **means for** a, b, c **being** Element **of the** L carrier **of it holds** a⊓(b⊔c) = (a⊓b)⊔(a⊓c).

Theorem LATTICES:11. (**for** a, b, c **holds** a⊓(b⊔c) = (a⊓b)⊔(a⊓c)) **implies** L **is** D Lattice.

Definition

mode M Lattice → Lattice **means** **for** a, b, c **being** Element **of the** L carrier **of it st** a ⊑ c **holds** a⊔(b⊓c) = (a⊔b)⊓c.

Theorem LATTICES:12. (**for** a, b, c **st** a ⊑ c **holds** a⊔(b⊓c) = (a⊔b)⊓c) **implies** L **is** M Lattice.

Definition

mode 0 Lattice → Lattice **means** **ex** c **being** Element **of the** L carrier **of it st for** a **being** Element **of the** L carrier **of it holds** c⊓a = c.

Theorem LATTICES:13. (**ex** c **st for** a **holds** c⊓a = c) **implies** L **is** 0 Lattice.

Definition

mode 1 Lattice → Lattice **means** **ex** c **being** Element **of the** L carrier **of it st for** a **being** Element **of the** L carrier **of it holds** c⊔a = c.

Theorem LATTICES:14. (**ex** c **st for** a **holds** c⊔a = c) **implies** L **is** 1 Lattice.

Definition

mode 01 Lattice → Lattice **means** **it is** 0 Lattice & **it is** 1 Lattice.

Theorem LATTICES:15. (L **is** 0 Lattice & L **is** 1 Lattice) **implies** L **is** 01 Lattice.

Definition

let L.

assume **ex** c **st for** a **holds** c⊓a = c.

func ⊥L → Element **of the** L carrier **of** L **means** **it**⊓a = **it**.

Definition

let L **be** 0 Lattice.

redefine

func ⊥L → Element **of the** L carrier **of** L.

Definition

let L.

assume **ex** c **st for** a **holds** c⊔a = c.

func ⊤L → Element **of the** L carrier **of** L **means** **it**⊔a = **it**.

Definition

let L **be** 1 Lattice.

redefine

func ⊤L → Element **of the** L carrier **of** L.

Definition

let L **be** 01 Lattice.

redefine

func ⊥L → Element **of the** L carrier **of** L.

func $\top$L $\to$ Element **of the** L carrier **of** L.

Definition

let L, a, b.

assume L **is** 01 Lattice.

pred a is a complement b **means** a$\sqcup$b = $\top$L & a$\sqcap$b = $\bot$L.

Definition

mode C Lattice $\to$ 01 Lattice **means** **for** b **being** Element **of the** L carrier **of it** **ex** a **being** Element **of the** L carrier **of it** **st** a is a complement b.

Definition

mode B Lattice $\to$ C Lattice **means** **it** **is** D Lattice.

Theorem LATTICES:16. a$\sqcup$b = b **iff** a$\sqcap$b = a.

Theorem LATTICES:17. a$\sqcup$a = a.

Theorem LATTICES:18. a$\sqcap$a = a.

Theorem LATTICES:19. **for** L **holds** (**for** a, b, c **holds** a$\sqcap$(b$\sqcup$c) = (a$\sqcap$b)$\sqcup$(a$\sqcap$c)) **iff** (**for** a, b, c **holds** a$\sqcup$(b$\sqcap$c) = (a$\sqcup$b)$\sqcap$(a$\sqcup$c)).

Theorem LATTICES:20. a $\sqsubseteq$ b **iff** a$\sqcup$b = b.

Theorem LATTICES:21. a $\sqsubseteq$ b **iff** a$\sqcap$b = a.

Theorem LATTICES:22. a $\sqsubseteq$ a$\sqcup$b.

Theorem LATTICES:23. a$\sqcap$b $\sqsubseteq$ a.

Theorem LATTICES:24. a $\sqsubseteq$ a.

Theorem LATTICES:25. a $\sqsubseteq$ b & b $\sqsubseteq$ c **implies** a $\sqsubseteq$ c.

Theorem LATTICES:26. a $\sqsubseteq$ b & b $\sqsubseteq$ a **implies** a = b.

Theorem LATTICES:27. a $\sqsubseteq$ b **implies** a$\sqcap$c $\sqsubseteq$ b$\sqcap$c.

Theorem LATTICES:28. a $\sqsubseteq$ b **implies** c$\sqcap$a $\sqsubseteq$ c$\sqcap$b.

Theorem LATTICES:29. (**for** a, b, c **holds** (a$\sqcap$b)$\sqcup$(b$\sqcap$c)$\sqcup$(c$\sqcap$a) = (a$\sqcup$b)$\sqcap$(b$\sqcup$c)$\sqcap$(c$\sqcup$a)) **implies** L **is** D Lattice.

reserve L **for** D Lattice.

reserve a, b, c **for** Element **of the** L carrier **of** L.

Theorem LATTICES:30. **for** L **holds** (**for** a, b, c **holds** a$\sqcap$(b$\sqcup$c) = (a$\sqcap$b)$\sqcup$(a$\sqcap$c)) & (**for** a, b, c **holds** (b$\sqcup$c)$\sqcap$a = (b$\sqcap$a)$\sqcup$(c$\sqcap$a)).

Theorem LATTICES:31. **for** L **holds** (**for** a, b, c **holds** a$\sqcup$(b$\sqcap$c) = (a$\sqcup$b)$\sqcap$(a$\sqcup$c)) & (**for** a, b, c **holds** (b$\sqcap$c)$\sqcup$a = (b$\sqcup$a)$\sqcap$(c$\sqcup$a)).

Theorem LATTICES:32. c$\sqcap$a = c$\sqcap$b & c$\sqcup$a = c$\sqcup$b **implies** a = b.

Theorem LATTICES:33. a$\sqcap$c = b$\sqcap$c & a$\sqcup$c = b$\sqcup$c **implies** a = b.

Theorem LATTICES:34. (a$\sqcup$b)$\sqcap$(b$\sqcup$c)$\sqcap$(c$\sqcup$a) = (a$\sqcap$b)$\sqcup$(b$\sqcap$c)$\sqcup$(c$\sqcap$a).

Theorem LATTICES:35. L **is** M Lattice.

reserve L **for** M Lattice.

reserve a, b, c **for** Element **of the** L carrier **of** L.

Theorem LATTICES:36. a $\sqsubseteq$ c **implies** a$\sqcup$(b$\sqcap$c) = (a$\sqcup$b)$\sqcap$c.

Theorem LATTICES:37. c $\sqsubseteq$ a **implies** a$\sqcap$(b$\sqcup$c) = (a$\sqcap$b)$\sqcup$c.

reserve L **for** 0 Lattice.

reserve a, b, c **for** Element **of the** L carrier **of** L.

Theorem LATTICES:38. **ex** c **st for** a **holds** c$\sqcap$a = c.

Theorem LATTICES:39. $\perp$L$\sqcup$a = a & a$\sqcup\perp$L = a.

Theorem LATTICES:40. $\perp$L$\sqcap$a = $\perp$L & a$\sqcap\perp$L = $\perp$L.

Theorem LATTICES:41. $\perp$L $\sqsubseteq$ a.

reserve L **for** 1 Lattice.

reserve a, b, c **for** Element **of the** L carrier **of** L.

Theorem LATTICES:42. **ex** c **st for** a **holds** c$\sqcup$a = c.

Theorem LATTICES:43. $\top$L$\sqcap$a = a & a$\sqcap\top$L = a.

Theorem LATTICES:44. $\top$L$\sqcup$a = $\top$L & a$\sqcup\top$L = $\top$L.

Theorem LATTICES:45. a $\sqsubseteq$ $\top$L.

reserve L **for** C Lattice.

reserve a, b, c **for** Element **of the** L carrier **of** L.

Theorem LATTICES:46. **ex** a **st** a is a complement b.

reserve L **for** Lattice.

reserve a, b, c **for** Element **of the** L carrier **of** L.

Definition

let L.

let x **be** Element **of the** L carrier **of** L.

assume L **is** B Lattice.

func x^c $\rightarrow$ Element **of the** L carrier **of** L **means it** is a complement x.

Definition

let L **be** B Lattice.

let x **be** Element **of the** L carrier **of** L.

redefine

func x^c $\rightarrow$ Element **of the** L carrier **of** L.

reserve L **for** B Lattice.

reserve a, b, c **for** Element **of the** L carrier **of** L.

Theorem LATTICES:47. $a^c$$\sqcap$a = $\perp$L & a$\sqcap$$a^c$ = $\perp$L.

Theorem LATTICES:48. $a^c \sqcup a = \top L$ & $a \sqcup a^c = \top L$.

Theorem LATTICES:49. $a^{cc} = a$.

Theorem LATTICES:50. $(a \sqcap b)^c = a^c \sqcup b^c$.

Theorem LATTICES:51. $(a \sqcup b)^c = a^c \sqcap b^c$.

Theorem LATTICES:52. $b \sqcap a = \bot L$ **iff** $b \sqsubseteq a^c$.

Theorem LATTICES:53. $a \sqsubseteq b$ **implies** $b^c \sqsubseteq a^c$.

Chapter 28

PRE_TOPC

Topological Spaces and Continuous Functions

by

Beata Padlewska [1]

Warsaw University (Białystok)

Agata Darmochwał[2]

Warsaw University (Białystok)

Summary. The article contains a definition of topological space. The following notions are defined: point of topological space, subset of topological space, subspace of topological space, and continuous function.

The symbols used in this article are introduced in the following vocabularies: BOOLE, FUNC, FUNC_REL, REAL_1, SUB_OP, FAM_OP, SFAMILY, and TOPCON. The terminology and notation used in this article have been introduced in the following articles: TARSKI, BOOLE, ENUMSET1, SUBSET_1, FUNCT_1, ORDINAL1, MCART_1, DOMAIN_1, FUNCT_2, and SETFAM_1.

struct TopStruct ⟨⟨carrier → DOMAIN, topology → Subset-Family **of the** carrier⟩⟩.

reserve T **for** TopStruct.

reserve p, q **for** Subset **of the** carrier **of** T.

[1]Supported by RPBP.III-24.C1.

[2]Supported by RPBP.III-24.C1.

reserve x **for** Any.

Definition

mode TopSpace → TopStruct **means** ∅ ∈ **the** topology **of it** & **the** carrier **of it** ∈ **the** topology **of it** & (**for** a **being** Subset-Family **of the** carrier **of it st** a ⊆ **the** topology **of it holds** ⋃a ∈ **the** topology **of it**) & (**for** a, b **being** Subset **of the** carrier **of it st** a ∈ **the** topology **of it** & b ∈ **the** topology **of it holds** a∩b ∈ **the** topology **of it**).

Theorem PRE_TOPC:1. (∅ ∈ **the** topology **of** T & **the** carrier **of** T ∈ **the** topology **of** T & (**for** a **being** Subset-Family **of the** carrier **of** T **st** a ⊆ **the** topology **of** T **holds** ⋃a ∈ **the** topology **of** T) & (**for** p, q **being** Subset **of the** carrier **of** T **st** p ∈ **the** topology **of** T & q ∈ **the** topology **of** T **holds** p∩q ∈ **the** topology **of** T)) **implies** T **is** TopSpace.

reserve T, S, GX, GY **for** TopSpace.

Definition

let T.

mode Point **of** T → Element **of the** carrier **of** T **means not contradiction**.

Theorem PRE_TOPC:2. **for** x **being** Element **of the** carrier **of** T **holds** x **is** Point **of** T.

Definition

let T.

mode Subset **of** T → **set of** Point **of** T **means not contradiction**.

Theorem PRE_TOPC:3. **for** P **being** Subset **of the** carrier **of** T **holds** P **is** Subset **of** T.

reserve P, Q, R **for** Subset **of** T.

reserve p, q, r **for** Point **of** T.

Definition

let T.

mode Subset-Family **of** T → Subset-Family **of the** carrier **of** T **means not contradiction**.

Theorem PRE_TOPC:4. **for** F **being** Subset-Family **of the** carrier **of** T **holds** F **is** Subset-Family **of** T.

reserve F **for** Subset-Family **of** T.

scheme SubFamEx1{A() → TopSpace, P[Subset **of** A()]}: **ex** F **being** Subset-Family **of** A() **st for** B **being** Subset **of** A() **holds** B ∈ F **iff** P[B].

Theorem PRE_TOPC:5. ∅ ∈ **the** topology **of** T.

Theorem PRE_TOPC:6. **the** carrier **of** T ∈ **the** topology **of** T.

Theorem PRE_TOPC:7. **for** a **being** Subset-Family **of** T **st** a ⊆ **the** topology **of** T **holds** ⋃a ∈ **the** topology **of** T.

Theorem PRE_TOPC:8. P $\in$ **the** topology **of** T & Q $\in$ **the** topology **of** T **implies** P$\cap$Q $\in$ **the** topology **of** T.

Definition

let T.

func $\emptyset$(T) $\to$ Subset **of** T **means it** = $\emptyset$ **the** carrier **of** T.

func Ω(T) $\to$ Subset **of** T **means it** = Ω**the** carrier **of** T.

Theorem PRE_TOPC:9. $\emptyset$ T = $\emptyset$ **the** carrier **of** T.

Theorem PRE_TOPC:10. ΩT = Ω**the** carrier **of** T.

Theorem PRE_TOPC:11. $\emptyset$(T) = $\emptyset$.

Theorem PRE_TOPC:12. Ω(T) = **the** carrier **of** T.

Definition

let T, P.

func P^c $\to$ Subset **of** T **means it** = P^c.

Definition

let T, P, Q.

redefine

func P$\cup$Q $\to$ Subset **of** T.

func P$\cap$Q $\to$ Subset **of** T.

func P$\setminus$Q $\to$ Subset **of** T.

func P$\dot{-}$Q $\to$ Subset **of** T.

Theorem PRE_TOPC:13. p $\in$ Ω(T).

Theorem PRE_TOPC:14. P $\subseteq$ Ω(T).

Theorem PRE_TOPC:15. P$\cap\Omega$(T) = P.

Theorem PRE_TOPC:16. **for** A **being** set **holds** A $\subseteq$ Ω(T) **implies** A **is** Subset **of** T.

Theorem PRE_TOPC:17. P^c = Ω(T)$\setminus$P.

Theorem PRE_TOPC:18. P$\cup P^c$ = Ω(T).

Theorem PRE_TOPC:19. P $\subseteq$ Q **iff** $Q^c \subseteq P^c$.

Theorem PRE_TOPC:20. P = P^{cc}.

Theorem PRE_TOPC:21. P $\subseteq$ Q^c **iff** P$\cap$Q = $\emptyset$.

Theorem PRE_TOPC:22. Ω(T)$\setminus$(Ω(T)$\setminus$P) = P.

Theorem PRE_TOPC:23. P $\neq$ Ω(T) **iff** Ω(T)$\setminus$P $\neq$ $\emptyset$.

Theorem PRE_TOPC:24. Ω(T)$\setminus$P = Q **implies** Ω(T) = P$\cup$Q.

Theorem PRE_TOPC:25. Ω(T) = P$\cup$Q & P$\cap$Q = $\emptyset$ **implies** Q = Ω(T)$\setminus$P.

Theorem PRE_TOPC:26. P$\cap P^c$ = $\emptyset$(T).

Theorem PRE_TOPC:27. $\Omega(T) = (\emptyset\ T)^c$.

Theorem PRE_TOPC:28. $P \setminus Q = P \cap Q^c$.

Theorem PRE_TOPC:29. P = Q **implies** $\Omega(T) \setminus P = \Omega(T) \setminus Q$.

Definition

let T, P.

pred P is open **means** P ∈ **the** topology **of** T.

Theorem PRE_TOPC:30. P is open **iff** P ∈ **the** topology **of** T.

Definition

let T, P.

pred P is closed **means** $\Omega(T) \setminus P$ is open.

Theorem PRE_TOPC:31. P is closed **iff** $\Omega(T) \setminus P$ is open.

Definition

let T, P.

pred P is open closed **means** P is open & P is closed.

Theorem PRE_TOPC:32. P is open closed **iff** P is open & P is closed.

Definition

let T, F.

redefine

func $\bigcup F \to$ Subset **of** T.

Definition

let T, F.

redefine

func $\bigcap F \to$ Subset **of** T.

Definition

let T, F.

pred F is a cover of T **means** $\Omega(T) = \bigcup F$.

Theorem PRE_TOPC:33. F is a cover of T **iff** $\Omega(T) = \bigcup F$.

Definition

let T.

mode SubSpace **of** T → TopSpace **means** Ω(**it**) ⊆ Ω(T) & **for** P **being** Subset **of it holds** P ∈ **the** topology **of it iff ex** Q **being** Subset **of** T **st** Q ∈ **the** topology **of** T & P = Q∩Ω(**it**).

Theorem PRE_TOPC:34. (Ω(S) ⊆ Ω(T) & **for** P **being** Subset **of** S **holds** P ∈ **the** topology **of** S **iff ex** Q **being** Subset **of** T **st** Q ∈ **the** topology **of** T & P = Q∩Ω(S)) **implies** S **is** SubSpace **of** T.

Theorem PRE_TOPC:35. **for** V **being** SubSpace **of** T **holds** Ω(V) ⊆ Ω(T) & **for** P **being** Subset **of** V **holds** P ∈ **the** topology **of** V **iff ex** Q **being** Subset **of** T **st** Q ∈ **the** topology **of** T & P = Q∩Ω(V).

Definition

let T, P.

assume P ≠ ∅(T).

func T↾P → SubSpace **of** T **means** Ω(**it**) = P & ∅(**it**) = ∅.

Theorem PRE_TOPC:36. P ≠ ∅(T) **implies** Ω(T↾P) = P & ∅(T↾P) = ∅.

Definition

let T, S.

mode map **of** T, S → Function **of** (**the** carrier **of** T), (**the** carrier **of** S) **means not contradiction**.

Theorem PRE_TOPC:37. **for** f **being** Function **of the** carrier **of** T, **the** carrier **of** S **holds** f **is** map **of** T, S.

reserve f, g **for** map **of** T, S.

reserve P1, Q1, R1 **for** Subset **of** S.

Definition

let T, S, f, P.

redefine

func f.P → (Subset **of** S).

Definition

let T, S, f, P1.

redefine

func f^{-1}P1 → (Subset **of** T).

Definition

let T, S, f.

pred f is continuous **means for** P1 **holds** P1 is closed **implies** f^{-1}P1 is closed.

Theorem PRE_TOPC:38. f is continuous **iff** (**for** P1 **holds** P1 is closed **implies** f^{-1}P1 is closed).

scheme TopAbstr{A() → TopSpace, P[Point **of** A()]}: **ex** P **being** Subset **of** A() **st for** x **being** Point **of** A() **holds** x ∈ P **iff** P[x].

Theorem PRE_TOPC:39. **for** X′ **being** SubSpace **of** GX **for** A **being** Subset **of** X′ **holds** A **is** Subset **of** GX.

Theorem PRE_TOPC:40. **for** A **being** (Subset **of** GX), x **being** Any **st** x ∈ A **holds** x **is** Point **of** GX.

Theorem PRE_TOPC:41. **for** A **being** Subset **of** GX **st** A ≠ ∅(GX) **ex** x **being** Point **of** GX **st** x ∈ A.

Theorem PRE_TOPC:42. $\Omega(\mathrm{GX})$ is closed.

Theorem PRE_TOPC:43. **for** X' **being** (SubSpace **of** GX), B **being** Subset **of** X' **holds** B is closed **iff ex** C **being** Subset **of** GX **st** C is closed & $\mathrm{C} \cap (\Omega(\mathrm{X}')) = \mathrm{B}$.

Theorem PRE_TOPC:44. **for** F **being** Subset-Family **of** GX **st** $\mathrm{F} \neq \emptyset$ & **for** A **being** Subset **of** GX **st** $\mathrm{A} \in \mathrm{F}$ **holds** A is closed **holds** $\bigcap \mathrm{F}$ is closed.

Definition

let GX **be** TopSpace, A **be** Subset **of** GX.

func Cl A $\rightarrow$ Subset **of** GX **means for** p **being** Point **of** GX **holds** $\mathrm{p} \in$ **it iff for** G **being** Subset **of** GX **st** G is open **holds** $\mathrm{p} \in \mathrm{G}$ **implies** $\mathrm{A} \cap \mathrm{G} \neq \emptyset(\mathrm{GX})$.

Theorem PRE_TOPC:45. **for** A **being** (Subset **of** GX), p **being** Point **of** GX **holds** $\mathrm{p} \in$ Cl A **iff for** C **being** Subset **of** GX **st** C is closed **holds** ($\mathrm{A} \subseteq \mathrm{C}$ **implies** $\mathrm{p} \in \mathrm{C}$).

Theorem PRE_TOPC:46. **for** A **being** (Subset **of** GX) **ex** F **being** Subset-Family **of** GX **st** (**for** C **being** Subset **of** GX **holds** $\mathrm{C} \in \mathrm{F}$ **iff** C is closed & $\mathrm{A} \subseteq \mathrm{C}$) & Cl A $= \bigcap \mathrm{F}$.

Theorem PRE_TOPC:47. **for** X' **being** (SubSpace **of** GX), A **being** (Subset **of** GX), A1 **being** Subset **of** X' **st** A = A1 **holds** Cl A1 = (Cl A)$\cap(\Omega(\mathrm{X}'))$.

Theorem PRE_TOPC:48. **for** A **being** Subset **of** GX **holds** $\mathrm{A} \subseteq$ Cl A.

Theorem PRE_TOPC:49. **for** A, B **being** Subset **of** GX **st** $\mathrm{A} \subseteq \mathrm{B}$ **holds** Cl A $\subseteq$ Cl B.

Theorem PRE_TOPC:50. **for** A, B **being** Subset **of** GX **holds** Cl ($\mathrm{A} \cup \mathrm{B}$) = Cl A$\cup$Cl B.

Theorem PRE_TOPC:51. **for** A, B **being** Subset **of** GX **holds** Cl ($\mathrm{A} \cap \mathrm{B}$) $\subseteq$ (Cl A)$\cap$Cl B.

Theorem PRE_TOPC:52. **for** A **being** Subset **of** GX **holds** A is closed **iff** Cl A = A.

Theorem PRE_TOPC:53. **for** A **being** Subset **of** GX **holds** A is open **iff** Cl ($\Omega(\mathrm{GX}) \setminus \mathrm{A}$) = $\Omega(\mathrm{GX}) \setminus \mathrm{A}$.

Theorem PRE_TOPC:54. **for** A **being** (Subset **of** GX), p **being** Point **of** GX **holds** $\mathrm{p} \in$ Cl A **iff for** G **being** Subset **of** GX **st** G is open **holds** $\mathrm{p} \in \mathrm{G}$ **implies** $\mathrm{A} \cap \mathrm{G} \neq \emptyset(\mathrm{GX})$.

Chapter 29

TOPS_1

Subsets of a Topological Space

by

Mirosław Wysocki [1]

Warsaw University (Białystok)

Agata Darmochwał[2]

Warsaw University (Białystok)

Summary. The article contains some theorems about open and closed sets. The following topological operations on sets are defined: closure, interior and frontier. The following notions are introduced: dense set, boundary set, nowheredense set and set being domain (closed domain and open domain), and some basic facts concerning them are proved.

The symbols used in this article are introduced in the following vocabularies: BOOLE, FUNC, FUNC_REL, REL_REL, REAL_1, SUB_OP, FAM_OP, SFAMILY, TOPCON, and TOP1. The terminology and notation used in this article have been introduced in the following articles: TARSKI, BOOLE, ENUMSET1, SUBSET_1, FUNCT_1, ORDINAL1, MCART_1, DOMAIN_1, FUNCT_2, SETFAM_1, and PRE_TOPC.

reserve TS **for** TopSpace.

reserve x **for** Any.

[1]Supported by RPBP.III-24.C1.

[2]Supported by RPBP.III-24.C1.

reserve X, Y, Z **for** set.

reserve P, Q, G **for** Subset **of** TS.

reserve p **for** Point **of** TS.

Theorem TOPS_1:1. x ∈ P **implies** x **is** Point **of** TS.

Theorem TOPS_1:2. $P \cup \Omega TS = \Omega TS$ & $\Omega TS \cup P = \Omega TS$.

Theorem TOPS_1:3. $P \cap \Omega TS = P$ & $\Omega TS \cap P = P$.

Theorem TOPS_1:4. $P \cap \emptyset$ TS = $\emptyset$ TS & $\emptyset$ TS$\cap$P = $\emptyset$ TS.

Theorem TOPS_1:5. $P^c = \Omega TS \setminus P$.

Theorem TOPS_1:6. P^c = (P **qua** Subset **of the** carrier **of** TS)^c.

Theorem TOPS_1:7. p ∈ P^c **iff not** p ∈ P.

Theorem TOPS_1:8. $(\Omega TS)^c = \emptyset$ TS.

Theorem TOPS_1:9. $\Omega TS = (\emptyset\ TS)^c$.

Theorem TOPS_1:10. $(P^c)^c = P$.

Theorem TOPS_1:11. $P \cup P^c = \Omega TS$ & $P^c \cup P = \Omega TS$.

Theorem TOPS_1:12. $P \cap P^c = \emptyset$ TS & $P^c \cap P = \emptyset$ TS.

Theorem TOPS_1:13. $(P \cup Q)^c = (P^c) \cap (Q^c)$.

Theorem TOPS_1:14. $(P \cap Q)^c = (P^c) \cup (Q^c)$.

Theorem TOPS_1:15. $P \subseteq Q$ **iff** $Q^c \subseteq P^c$.

Theorem TOPS_1:16. $P \setminus Q = P \cap Q^c$.

Theorem TOPS_1:17. $(P \setminus Q)^c = P^c \cup Q$.

Theorem TOPS_1:18. $P \subseteq Q^c$ **implies** $Q \subseteq P^c$.

Theorem TOPS_1:19. $P^c \subseteq Q$ **implies** $Q^c \subseteq P$.

Theorem TOPS_1:20. $P \subseteq Q$ **iff** $P \cap Q^c = \emptyset$.

Theorem TOPS_1:21. $P^c = Q^c$ **implies** $P = Q$.

Theorem TOPS_1:22. $\emptyset$ TS is closed.

Theorem TOPS_1:23. Cl ($\emptyset$ TS) = $\emptyset$ TS.

Theorem TOPS_1:24. P ⊆ Cl P.

Theorem TOPS_1:25. P ⊆ Q **implies** Cl P ⊆ Cl Q.

Theorem TOPS_1:26. Cl (Cl P) = Cl P.

Theorem TOPS_1:27. Cl (ΩTS) = ΩTS.

Theorem TOPS_1:28. ΩTS is closed.

Theorem TOPS_1:29. P is closed **iff** P^c is open.

Theorem TOPS_1:30. P is open **iff** P^c is closed.

Theorem TOPS_1:31. Q is closed & P ⊆ Q **implies** Cl P ⊆ Q.

Theorem TOPS_1:32. Cl P$\setminus$Cl Q ⊆ Cl (P$\setminus$Q).

Theorem TOPS_1:33. Cl (P∩Q) ⊆ Cl P∩Cl Q.

Theorem TOPS_1:34. P is closed & Q is closed **implies** Cl (P∩Q) = Cl P∩Cl Q.

Theorem TOPS_1:35. P is closed & Q is closed **implies** P∩Q is closed.

Theorem TOPS_1:36. P is closed & Q is closed **implies** P∪Q is closed.

Theorem TOPS_1:37. P is open & Q is open **implies** P∪Q is open.

Theorem TOPS_1:38. P is open & Q is open **implies** P∩Q is open.

Theorem TOPS_1:39. p ∈ Cl P **iff for** G **st** G is open **holds** (p ∈ G **implies** P∩G ≠ ∅).

Theorem TOPS_1:40. Q is open **implies** Q∩Cl P ⊆ Cl (Q∩P).

Theorem TOPS_1:41. Q is open **implies** Cl (Q∩Cl P) = Cl (Q∩P).

Definition

let TS, P.

func Int P → Subset **of** TS **means it** = (Cl (P^c))^c.

Theorem TOPS_1:42. Int P = (Cl P^c)^c.

Theorem TOPS_1:43. Int (ΩTS) = ΩTS.

Theorem TOPS_1:44. Int P ⊆ P.

Theorem TOPS_1:45. Int (Int P) = Int P.

Theorem TOPS_1:46. Int P∩Int Q = Int (P∩Q).

Theorem TOPS_1:47. Int (∅ TS) = ∅ TS.

Theorem TOPS_1:48. P ⊆ Q **implies** Int P ⊆ Int Q.

Theorem TOPS_1:49. Int P∪Int Q ⊆ Int (P∪Q).

Theorem TOPS_1:50. Int (P∖Q) ⊆ Int P∖Int Q.

Theorem TOPS_1:51. Int P is open.

Theorem TOPS_1:52. ∅ TS is open.

Theorem TOPS_1:53. ΩTS is open.

Theorem TOPS_1:54. x ∈ Int P **iff ex** Q **st** Q is open & Q ⊆ P & x ∈ Q.

Theorem TOPS_1:55. P is open **iff** Int P = P.

Theorem TOPS_1:56. Q is open & Q ⊆ P **implies** Q ⊆ Int P.

Theorem TOPS_1:57. P is open **iff** (**for** x **holds** x ∈ P **iff ex** Q **st** Q is open & Q ⊆ P & x ∈ Q).

Theorem TOPS_1:58. Cl (Int P) = Cl (Int (Cl (Int P))).

Theorem TOPS_1:59. P is open **implies** Cl (Int (Cl P)) = Cl P.

Definition

let TS, P.

func Fr P → Subset **of** TS **means it** = Cl P∩Cl (P^c).

Theorem TOPS_1:60. $\mathsf{Fr}\ P = \mathsf{Cl}\ P \cap \mathsf{Cl}\ (P^c)$.

Theorem TOPS_1:61. $p \in \mathsf{Fr}\ P$ **iff** (**for** Q **st** Q is open & $p \in Q$ **holds** ($P \cap Q \neq \emptyset$ & $P^c \cap Q \neq \emptyset$)).

Theorem TOPS_1:62. $\mathsf{Fr}\ P = \mathsf{Fr}\ (P^c)$.

Theorem TOPS_1:63. $\mathsf{Fr}\ P \subseteq \mathsf{Cl}\ P$.

Theorem TOPS_1:64. $\mathsf{Fr}\ P = \mathsf{Cl}\ (P^c) \cap P \cup (\mathsf{Cl}\ P \setminus P)$.

Theorem TOPS_1:65. $\mathsf{Cl}\ P = P \cup \mathsf{Fr}\ P$.

Theorem TOPS_1:66. $\mathsf{Fr}\ (P \cap Q) \subseteq \mathsf{Fr}\ P \cup \mathsf{Fr}\ Q$.

Theorem TOPS_1:67. $\mathsf{Fr}\ (P \cup Q) \subseteq \mathsf{Fr}\ P \cup \mathsf{Fr}\ Q$.

Theorem TOPS_1:68. $\mathsf{Fr}\ (\mathsf{Fr}\ P) \subseteq \mathsf{Fr}\ P$.

Theorem TOPS_1:69. P is closed **implies** $\mathsf{Fr}\ P \subseteq P$.

Theorem TOPS_1:70. $\mathsf{Fr}\ P \cup \mathsf{Fr}\ Q = \mathsf{Fr}\ (P \cup Q) \cup \mathsf{Fr}\ (P \cap Q) \cup (\mathsf{Fr}\ P \cap \mathsf{Fr}\ Q)$.

Theorem TOPS_1:71. $\mathsf{Fr}\ (\mathsf{Int}\ P) \subseteq \mathsf{Fr}\ P$.

Theorem TOPS_1:72. $\mathsf{Fr}\ (\mathsf{Cl}\ P) \subseteq \mathsf{Fr}\ P$.

Theorem TOPS_1:73. $\mathsf{Int}\ P \cap \mathsf{Fr}\ P = \emptyset$.

Theorem TOPS_1:74. $\mathsf{Int}\ P = P \setminus \mathsf{Fr}\ P$.

Theorem TOPS_1:75. $\mathsf{Fr}\ (\mathsf{Fr}\ (\mathsf{Fr}\ P)) = \mathsf{Fr}\ (\mathsf{Fr}\ P)$.

Theorem TOPS_1:76. P is open **iff** $\mathsf{Fr}\ P = \mathsf{Cl}\ P \setminus P$.

Theorem TOPS_1:77. P is closed **iff** $\mathsf{Fr}\ P = P \setminus \mathsf{Int}\ P$.

Definition

let TS, P.

pred P is dense **means** $\mathsf{Cl}\ P = \Omega TS$.

Theorem TOPS_1:78. P is dense **iff** $\mathsf{Cl}\ P = \Omega TS$.

Theorem TOPS_1:79. P is dense & $P \subseteq Q$ **implies** Q is dense.

Theorem TOPS_1:80. P is dense **iff** (**for** Q **st** $Q \neq \emptyset$ & Q is open **holds** $P \cap Q \neq \emptyset$).

Theorem TOPS_1:81. P is dense **implies** (**for** Q **holds** Q is open **implies** $\mathsf{Cl}\ Q = \mathsf{Cl}\ (Q \cap P)$).

Theorem TOPS_1:82. P is dense & Q is dense & Q is open **implies** $P \cap Q$ is dense.

Definition

let TS, P.

pred P is boundary **means** P^c is dense.

Theorem TOPS_1:83. P is boundary **iff** P^c is dense.

Theorem TOPS_1:84. P is boundary **iff** $\mathsf{Int}\ P = \emptyset$.

Theorem TOPS_1:85. P is boundary & Q is boundary & Q is closed **implies** $P \cup Q$ is boundary.

Theorem TOPS_1:86. P is boundary **iff** (**for** Q **st** Q $\subseteq$ P & Q is open **holds** Q = $\emptyset$).

Theorem TOPS_1:87. 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 = $\emptyset$).

Theorem TOPS_1:88. P is boundary **iff** P $\subseteq$ Fr P.

Definition

let TS, P.

pred P is nowheredense **means** Cl P is boundary.

Theorem TOPS_1:89. P is nowheredense **iff** Cl P is boundary.

Theorem TOPS_1:90. P is nowheredense & Q is nowheredense **implies** P$\cup$Q is nowheredense.

Theorem TOPS_1:91. P is nowheredense **implies** P^c is dense.

Theorem TOPS_1:92. P is nowheredense **implies** P is boundary.

Theorem TOPS_1:93. Q is boundary & Q is closed **implies** Q is nowheredense.

Theorem TOPS_1:94. P is closed **implies** (P is nowheredense **iff** P = Fr P).

Theorem TOPS_1:95. P is open **implies** Fr P is nowheredense.

Theorem TOPS_1:96. P is closed **implies** Fr P is nowheredense.

Theorem TOPS_1:97. P is open & P is nowheredense **implies** P = $\emptyset$.

Definition

let TS, P.

pred P is domain **means** Int (Cl P) $\subseteq$ P & P $\subseteq$ Cl (Int P).

pred P is closed domain **means** P = Cl (Int P).

pred P is open domain **means** P = Int (Cl P).

Theorem TOPS_1:98. P is domain **iff** Int (Cl P) $\subseteq$ P & P $\subseteq$ Cl (Int P).

Theorem TOPS_1:99. P is closed domain **iff** P = Cl (Int P).

Theorem TOPS_1:100. P is open domain **iff** P = Int (Cl P).

Theorem TOPS_1:101. P is open domain **iff** P^c is closed domain.

Theorem TOPS_1:102. P is closed domain **implies** Fr (Int P) = Fr P.

Theorem TOPS_1:103. P is closed domain **implies** Fr P $\subseteq$ Cl (Int P).

Theorem TOPS_1:104. P is open domain **implies** Fr P = Fr (Cl P) & Fr (Cl P) = Cl P$\setminus$P.

Theorem TOPS_1:105. P is open & P is closed **implies** (P is closed domain **iff** P is open domain).

Theorem TOPS_1:106. P is closed & P is domain **iff** P is closed domain.

Theorem TOPS_1:107. P is open & P is domain **iff** P is open domain.

Theorem TOPS_1:108. P is closed domain & Q is closed domain **implies** P$\cup$Q is closed domain.

Theorem TOPS_1:109. P is open domain & Q is open domain **implies** $P \cap Q$ is open domain.

Theorem TOPS_1:110. P is domain **implies** Int (Fr P) $= \emptyset$.

Theorem TOPS_1:111. P is domain **implies** Int P is domain & Cl P is domain.

Chapter 30

CONNSP_1

Connected Spaces

by

Beata Padlewska [1]

Warsaw University (Białystok)

Summary. The following notions are defined: separated sets, connected spaces, connected sets, components of a topological space, the component of a point. The definition of the boundary of a set is also included. The singleton of a point of a topological space is redefined as a subset of the space. Some theorems about these notions are proved.

The symbols used in this article are introduced in the following vocabularies: BOOLE, REAL_1, FUNC, FUNC_REL, REL_REL, SUB_OP, FAM_OP, SFAMILY, and TOPCON. The terminology and notation used in this article have been introduced in the following articles: TARSKI, BOOLE, ENUMSET1, FUNCT_1, SUBSET_1, SETFAM_1, ORDINAL1, MCART_1, DOMAIN_1, FUNCT_2, PRE_TOPC, and TOPS_1.

reserve GX, GY **for** TopSpace.

reserve A, A1, B, B1, C **for** Subset **of** GX.

Definition

let GX **be** TopSpace, A, B **be** Subset **of** GX.

pred A, B are separated **means** Cl A∩B = ∅(GX) & A∩Cl B = ∅(GX).

[1]Supported by RPBP.III-24.C1.

Theorem CONNSP_1:1. A, B are separated **implies** B, A are separated.

Theorem CONNSP_1:2. A, B are separated **implies** $A \cap B = \emptyset(GX)$.

Theorem CONNSP_1:3. $\Omega(GX) = A \cup B$ & A is closed & B is closed & $A \cap B = \emptyset(GX)$ **implies** A, B are separated.

Theorem CONNSP_1:4. $\Omega(GX) = A \cup B$ & A is open & B is open & $A \cap B = \emptyset(GX)$ **implies** A, B are separated.

Theorem CONNSP_1:5. $\Omega(GX) = A \cup B$ & A, B are separated **implies** A is open closed & B is open closed.

Theorem CONNSP_1:6. **for** X' **being** (SubSpace **of** GX), P1, Q1 **being** (Subset **of** GX), P, Q **being** Subset **of** X' **st** P = P1 & Q = Q1 **holds** P, Q are separated **implies** P1, Q1 are separated.

Theorem CONNSP_1:7. **for** X' **being** (SubSpace **of** GX), P, Q **being** (Subset **of** GX), P1, Q1 **being** Subset **of** X' **st** P = P1 & Q = Q1 & $P \cup Q \subseteq \Omega(X')$ **holds** P, Q are separated **implies** P1, Q1 are separated.

Theorem CONNSP_1:8. A, B are separated & $A1 \subseteq A$ & $B1 \subseteq B$ **implies** A1, B1 are separated.

Theorem CONNSP_1:9. A, B are separated & A, C are separated **implies** A, $B \cup C$ are separated.

Theorem CONNSP_1:10. (A is closed & B is closed) **or** (A is open & B is open) **implies** $A \setminus B$, $B \setminus A$ are separated.

Definition

let GX **be** TopSpace.

pred GX is connected **means for** A, B **being** Subset **of** GX **st** $\Omega(GX) = A \cup B$ & A, B are separated **holds** $A = \emptyset(GX)$ **or** $B = \emptyset(GX)$.

Theorem CONNSP_1:11. GX is connected **iff for** A, B **being** Subset **of** GX **st** $\Omega(GX) = A \cup B$ & $A \neq \emptyset(GX)$ & $B \neq \emptyset(GX)$ & A is closed & B is closed **holds** $A \cap B \neq \emptyset(GX)$.

Theorem CONNSP_1:12. GX is connected **iff for** A, B **being** Subset **of** GX **st** $\Omega(GX) = A \cup B$ & $A \neq \emptyset(GX)$ & $B \neq \emptyset(GX)$ & A is open & B is open **holds** $A \cap B \neq \emptyset(GX)$.

Theorem CONNSP_1:13. GX is connected **iff for** A **being** Subset **of** GX **st** $A \neq \emptyset(GX)$ & $A \neq \Omega(GX)$ **holds** $(\mathrm{Cl}\ A) \cap \mathrm{Cl}\ (\Omega(GX) \setminus A) \neq \emptyset(GX)$.

Theorem CONNSP_1:14. GX is connected **iff for** A **being** Subset **of** GX **st** A is open closed **holds** $A = \emptyset(GX)$ **or** $A = \Omega(GX)$.

Theorem CONNSP_1:15. **for** F **being** map **of** GX, GY **st** F is continuous & $F.(\Omega(GX)) = \Omega(GY)$ & GX is connected **holds** GY is connected.

Definition

let GX **be** TopSpace, A **be** Subset **of** GX.

pred A is connected **means** $GX \restriction A$ is connected.

Theorem CONNSP_1:16. A $\neq \emptyset$(GX) **implies** (A is connected **iff for** P, Q **being** Subset **of** GX **st** A = P$\cup$Q & P, Q are separated **holds** P = $\emptyset$(GX) **or** Q = $\emptyset$(GX)).

Theorem CONNSP_1:17. A is connected & A $\subseteq$ B$\cup$C & B, C are separated **implies** A $\subseteq$ B **or** A $\subseteq$ C.

Theorem CONNSP_1:18. A is connected & B is connected & **not** A, B are separated **implies** A$\cup$B is connected.

Theorem CONNSP_1:19. C $\neq \emptyset$(GX) & C is connected & C $\subseteq$ A & A $\subseteq$ Cl C **implies** A is connected.

Theorem CONNSP_1:20. A $\neq \emptyset$(GX) & A is connected **implies** Cl A is connected.

Theorem CONNSP_1:21. GX is connected & A $\neq \emptyset$(GX) & A is connected & Ω(GX) $\setminus$A = B$\cup$C & B, C are separated **implies** A$\cup$B is connected & A$\cup$C is connected.

Theorem CONNSP_1:22. Ω(GX)$\setminus$A = B$\cup$C & B, C are separated & A is closed **implies** A$\cup$B is closed & A$\cup$C is closed.

Theorem CONNSP_1:23. C is connected & C$\cap$A $\neq \emptyset$(GX) & C$\setminus$A $\neq \emptyset$(GX) **implies** C$\cap$Fr A $\neq \emptyset$(GX).

Theorem CONNSP_1:24. **for** X$'$ **being** (SubSpace **of** GX), A **being** (Subset **of** GX), B **being** Subset **of** X$'$ **st** A $\neq \emptyset$(GX) & A = B **holds** A is connected **iff** B is connected.

Theorem CONNSP_1:25. A$\cap$B $\neq \emptyset$(GX) & A is closed & B is closed **implies** (A$\cup$B is connected & A$\cap$B is connected **implies** A is connected & B is connected).

Theorem CONNSP_1:26. **for** F **being** Subset-Family **of** GX **st** (**for** A **being** Subset **of** GX **st** A $\in$ F **holds** A is connected) & (**ex** A **being** Subset **of** GX **st** A $\neq \emptyset$(GX) & A $\in$ F & (**for** B **being** Subset **of** GX **st** B $\in$ F & B $\neq$ A **holds not** A, B are separated)) **holds** $\bigcup$F is connected.

Theorem CONNSP_1:27. **for** F **being** Subset-Family **of** GX **st** (**for** A **being** Subset **of** GX **st** A $\in$ F **holds** A is connected) & $\bigcap$F $\neq \emptyset$(GX) **holds** $\bigcup$F is connected.

Theorem CONNSP_1:28. Ω(GX) is connected **iff** GX is connected.

Definition

let GX **be** TopSpace, x **be** Point **of** GX.

redefine

func {x} $\rightarrow$ Subset **of** GX.

Theorem CONNSP_1:29. **for** x **being** Point **of** GX **holds** {x} is connected.

Definition

let GX **be** TopSpace, x, y **be** Point **of** GX.

pred x, y are joined **means ex** C **being** Subset **of** GX **st** C is connected & x $\in$ C & y $\in$ C.

Theorem CONNSP_1:30. (**ex** x **being** Point **of** GX **st for** y **being** Point **of** GX **holds** x, y are joined) **implies** GX is connected.

Theorem CONNSP_1:31. (**ex** x **being** Point **of** GX **st** **for** y **being** Point **of** GX **holds** x, y are joined) **iff** (**for** x, y **being** Point **of** GX **holds** x, y are joined).

Theorem CONNSP_1:32. (**for** x, y **being** Point **of** GX **holds** x, y are joined) **implies** GX is connected.

Theorem CONNSP_1:33. **for** x **being** (Point **of** GX), F **being** Subset-Family **of** GX **st** **for** A **being** Subset **of** GX **holds** A ∈ F **iff** A is connected & x ∈ A **holds** F ≠ ∅.

Definition

let GX **be** TopSpace, A **be** Subset **of** GX.

pred A is a component of GX **means** A is connected & **for** B **being** Subset **of** GX **st** B is connected **holds** A ⊆ B **implies** A = B.

Theorem CONNSP_1:34. A is a component of GX **implies** A ≠ ∅(GX).

Theorem CONNSP_1:35. A is a component of GX **implies** A is closed.

Theorem CONNSP_1:36. A is a component of GX & B is a component of GX **implies** A = B **or** (A ≠ B **implies** A, B are separated).

Theorem CONNSP_1:37. A is a component of GX & B is a component of GX **implies** A = B **or** (A ≠ B **implies** A∩B = ∅(GX)).

Theorem CONNSP_1:38. C is connected **implies** **for** S **being** Subset **of** GX **st** S is a component of GX **holds** C∩S = ∅(GX) **or** C ⊆ S.

Definition

let GX **be** TopSpace, A, B **be** Subset **of** GX.

pred B is a component of A **means** **ex** B1 **being** Subset **of** GX↾A **st** B1 = B & B1 is a component of (GX↾A).

Theorem CONNSP_1:39. GX is connected & A ≠ Ω(GX) & A ≠ ∅(GX) & A is connected & C is a component of (Ω(GX)∖A) **implies** (Ω(GX)∖C) is connected.

Definition

let GX **be** TopSpace, x **be** Point **of** GX.

func skl x → Subset **of** GX **means** **ex** F **being** Subset-Family **of** GX **st** (**for** A **being** Subset **of** GX **holds** A ∈ F **iff** A is connected & x ∈ A) & ⋃F = **it**.

reserve x, y **for** Point **of** GX.

Theorem CONNSP_1:40. x ∈ skl x.

Theorem CONNSP_1:41. skl x is connected.

Theorem CONNSP_1:42. C is connected **implies** (skl x ⊆ C **implies** C = skl x).

Theorem CONNSP_1:43. A is a component of GX **iff** **ex** x **being** Point **of** GX **st** A = skl x.

Theorem CONNSP_1:44. A is a component of GX & x ∈ A **implies** A = skl x.

Theorem CONNSP_1:45. **for** S **being** Subset **of** GX **st** S = skl x **holds** (**for** p **being** Point **of** GX **st** p ≠ x & p ∈ S **holds** skl p = S).

Theorem CONNSP_1:46. **for** F **being** Subset-Family **of** GX **st** **for** A **being** Subset **of** GX **holds** A $\in$ F **iff** A is a component of GX **holds** F is a cover of GX.

Theorem CONNSP_1:47. A, B are separated **iff** Cl A$\cap$B = $\emptyset$(GX) & A$\cap$Cl B = $\emptyset$(GX).

Theorem CONNSP_1:48. GX is connected **iff** **for** A, B **being** Subset **of** GX **st** Ω(GX) = A$\cup$B & A, B are separated **holds** A = $\emptyset$(GX) **or** B = $\emptyset$(GX).

Theorem CONNSP_1:49. A is connected **iff** GX$\restriction$A is connected.

Theorem CONNSP_1:50. A is a component of GX **iff** A is connected & **for** B **being** Subset **of** GX **st** B is connected **holds** A $\subseteq$ B **implies** A = B.

Theorem CONNSP_1:51. B is a component of A **iff** **ex** B1 **being** Subset **of** GX$\restriction$A **st** B1 = B & B1 is a component of (GX$\restriction$A).

Theorem CONNSP_1:52. B = skl x **iff** **ex** F **being** Subset-Family **of** GX **st** (**for** A **being** Subset **of** GX **holds** A $\in$ F **iff** A is connected & x $\in$ A) & $\bigcup$F = B.

Chapter 31

SCHEMS_1

Some Basic Properties of Quantifiers

by

Stanisław T. Czuba [1]

Warsaw University (Białystok)

Summary. A number of schemes corresponding to simple tautologies of quantifier calculus are presented.

This article is written in plain Mizar; no additional vocabularies or signatures are referenced.

reserve a, b, c, d **for** Any.

scheme Schemat0{P[Any]}: **ex** a **st** P[a] **provided** A: **for** a **holds** P[a].

scheme Schemat1a{P[Any], T[]}: (**for** a **holds** P[a]) & T[] **provided** A: **for** a **holds** (P[a] & T[]).

scheme Schemat1b{P[Any], T[]}: **for** a **holds** (P[a] & T[]) **provided** A: (**for** a **holds** P[a]) & T[].

scheme Schemat2a{P[Any], T[]}: (**ex** a **st** P[a]) **or** T[] **provided** A: **ex** a **st** (P[a] **or** T[]).

scheme Schemat2b{P[Any], T[]}: **ex** a **st** (P[a] **or** T[]) **provided** A: (**ex** a **st** P[a]) **or** T[].

[1] Supported by RPBP.III-24.C1.

scheme Schemat3{S[Any, Any]}: **for** b **ex** a **st** S[a, b] **provided** A: **ex** a **st** **for** b **holds** S[a, b].

scheme Schemat4a{P[Any], Q[Any]}: (**ex** a **st** P[a]) **or** (**ex** a **st** Q[a]) **provided** A: **ex** a **st** (P[a] **or** Q[a]).

scheme Schemat4b{P[Any], Q[Any]}: **ex** a **st** (P[a] **or** Q[a]) **provided** A: (**ex** a **st** P[a]) **or** (**ex** a **st** Q[a]).

scheme Schemat5{P[Any], Q[Any]}: (**ex** a **st** P[a]) & (**ex** a **st** Q[a]) **provided** A: **ex** a **st** (P[a] & Q[a]).

scheme Schemat6a{P[Any], Q[Any]}: (**for** a **holds** P[a]) & (**for** a **holds** Q[a]) **provided** A: **for** a **holds** (P[a] & Q[a]).

scheme Schemat6b{P[Any], Q[Any]}: **for** a **holds** (P[a] & Q[a]) **provided** A: (**for** a **holds** P[a]) & (**for** a **holds** Q[a]).

scheme Schemat7{P[Any], Q[Any]}: **for** a **holds** (P[a] **or** Q[a]) **provided** A: (**for** a **holds** P[a]) **or** (**for** a **holds** Q[a]).

scheme Schemat8{P[Any], Q[Any]}: (**for** a **holds** P[a]) **implies** (**for** a **holds** Q[a]) **provided** A: **for** a **holds** P[a] **implies** Q[a].

scheme Schemat9{P[Any], Q[Any]}: (**for** a **holds** P[a]) **iff** (**for** a **holds** Q[a]) **provided** A: **for** a **holds** (P[a] **iff** Q[a]).

scheme Schemat10a{T[]}: T[] **provided** A: **for** a **holds** T[].

scheme Schemat10b{T[]}: **for** a **holds** T[] **provided** A: T[].

scheme Schemat11a{P[Any], T[]}: (**for** a **holds** P[a]) **or** T[] **provided** A: **for** a **holds** (P[a] **or** T[]).

scheme Schemat11b{P[Any], T[]}: **for** a **holds** (P[a] **or** T[]) **provided** A: (**for** a **holds** P[a]) **or** T[].

scheme Schemat12a{P[Any], T[]}: **ex** a **st** (T[] & P[a]) **provided** A: T[] & (**ex** a **st** P[a]).

scheme Schemat12b{P[Any], T[]}: T[] & (**ex** a **st** P[a]) **provided** A: **ex** a **st** (T[] & P[a]).

scheme Schemat13a{P[Any], T[]}: **for** a **holds** (T[] **implies** P[a]) **provided** A: T[] **implies** (**for** a **holds** P[a]).

scheme Schemat13b{P[Any], T[]}: T[] **implies** (**for** a **holds** P[a]) **provided** A: **for** a **holds** (T[] **implies** P[a]).

scheme Schemat14{P[Any], T[]}: **ex** a **st** (T[] **implies** P[a]) **provided** A: T[] **implies** (**ex** a **st** P[a]).

scheme Schemat15{P[Any], T[]}: **for** a **holds** (P[a] **implies** T[]) **provided** A: (**ex** a **st** P[a]) **implies** T[].

scheme Schemat16{P[Any], T[]}: **ex** a **st** (P[a] **implies** T[]) **provided** A: (**for** a **holds** P[a]) **implies** T[].

scheme Schemat17{P[Any], T[]}: (**for** a **holds** P[a]) **implies** T[] **provided** A: **for** a **holds** (P[a] **implies** T[]).

scheme Schemat18a{P[Any], Q[Any]}: **ex** a **st** (**for** b **holds** (P[a] **or** Q[b])) **provided** A: (**ex** a **st** P[a]) **or** (**for** b **holds** Q[b]).

scheme Schemat18b{P[Any], Q[Any]}: (**ex** a **st** P[a]) **or** (**for** b **holds** Q[b]) **provided** A: **ex** a **st** (**for** b **holds** (P[a] **or** Q[b])).

scheme Schemat19a{P[Any], Q[Any]}: **for** b **holds** (**ex** a **st** (P[a] **or** Q[b])) **provided** A: (**ex** a **st** P[a]) **or** (**for** b **holds** Q[b]).

scheme Schemat19b{P[Any], Q[Any]}: (**ex** a **st** P[a]) **or** (**for** b **holds** Q[b]) **provided** A: **for** b **holds** (**ex** a **st** (P[a] **or** Q[b])).

scheme Schemat20a{P[Any], Q[Any]}: **for** b **ex** a **st** (P[a] **or** Q[b]) **provided** A: **ex** a **st** (**for** b **holds** (P[a] **or** Q[b])).

scheme Schemat20b{P[Any], Q[Any]}: **ex** a **st** (**for** b **holds** (P[a] **or** Q[b])) **provided** A: **for** b **ex** a **st** (P[a] **or** Q[b]).

scheme Schemat21a{P[Any], Q[Any]}: **ex** a **st** **for** b **holds** P[a] & Q[b] **provided** A: (**ex** a **st** P[a]) & (**for** b **holds** Q[b]).

scheme Schemat21b{P[Any], Q[Any]}: (**ex** a **st** P[a]) & (**for** b **holds** Q[b]) **provided** A: **ex** a **st** **for** b **holds** P[a] & Q[b].

scheme Schemat22a{P[Any], Q[Any]}: **for** b **ex** a **st** (P[a] & Q[b]) **provided** A: (**ex** a **st** P[a]) & (**for** b **holds** Q[b]).

scheme Schemat22b{P[Any], Q[Any]}: (**ex** a **st** P[a]) & (**for** b **holds** Q[b]) **provided** A: **for** b **ex** a **st** (P[a] & Q[b]).

scheme Schemat23a{P[Any], Q[Any]}: **for** b **ex** a **st** P[a] & Q[b] **provided** A: **ex** a **st** **for** b **holds** P[a] & Q[b].

scheme Schemat23b{P[Any], Q[Any]}: **ex** a **st** **for** b **holds** (P[a] & Q[b]) **provided** A: **for** b **ex** a **st** (P[a] & Q[b]).

scheme Schemat24a{S[Any, Any], Q[Any]}: **for** a **ex** b **st** (S[a, b] **implies** Q[a]) **provided** A: **for** a **holds** ((**for** b **holds** S[a, b]) **implies** Q[a]).

scheme Schemat24b{S[Any, Any], Q[Any]}: **for** a **holds** ((**for** b **holds** S[a, b]) **implies** Q[a]) **provided** A: **for** a **ex** b **st** (S[a, b] **implies** Q[a]).

scheme Schemat25a{S[Any, Any], Q[Any]}: **for** a, b **holds** (S[a, b] **implies** Q[a]) **provided** A: **for** a **holds** ((**ex** b **st** S[a, b]) **implies** Q[a]).

scheme Schemat25b{S[Any, Any], Q[Any]}: **for** a **holds** ((**ex** b **st** S[a, b]) **implies** Q[a]) **provided** A: **for** a, b **holds** (S[a, b] **implies** Q[a]).

scheme Schemat26{S[Any, Any]}: **ex** a **st** **for** b **holds** S[a, b] **provided** A: **for** a, b **holds** S[a, b].

scheme Schemat27{S[Any, Any]}: **for** a **holds** S[a, a] **provided** A: **for** a, b **holds** S[a, b].

scheme Schemat28{S[Any, Any]}: **ex** b **st** **for** a **holds** S[a, b] **provided** A: **for** a, b **holds** S[a, b].

scheme Schemat29{S[Any, Any]}: **for** b **ex** a **st** S[a, b] **provided** A: **ex** a **st** **for** b **holds** S[a, b].

scheme Schemat30{S[Any, Any]}: **ex** a **st** S[a, a] **provided** A: **ex** a **st** **for** b **holds** S[a, b].

scheme Schemat31{S[Any, Any]}: **for** a **ex** b **st** S[b, a] **provided** A: **for** a **holds** S[a, a].

scheme Schemat32{S[Any, Any]}: **ex** a **st** S[a, a] **provided** A: **for** a **holds** S[a, a].

scheme Schemat33{S[Any, Any]}: **for** a **ex** b **st** S[a, b] **provided** A: **for** a **holds** S[a, a].

scheme Schemat34{S[Any, Any]}: **ex** b **st** S[b, b] **provided** A: **ex** b **st** **for** a **holds** S[a, b].

scheme Schemat35{S[Any, Any]}: **for** a **ex** b **st** S[a, b] **provided** A: **ex** b **st** **for** a **holds** S[a, b].

scheme Schemat36{S[Any, Any]}: **ex** a, b **st** S[a, b] **provided** A: **for** b **ex** a **st** S[a, b].

scheme Schemat37{S[Any, Any]}: **ex** a, b **st** S[a, b] **provided** A: **ex** a **st** S[a, a].

scheme Schemat38{S[Any, Any]}: **ex** a, b **st** S[a, b] **provided** A: **for** a **ex** b **st** S[a, b].

Chapter 32

ZF_LANG

A Model of ZF Set Theory Language

by

Grzegorz Bancerek [1]

Warsaw University (Białystok)

Summary. The goal of this article is to construct a language of the ZF set theory and to develop a notational and conceptual base which facilitates a convenient usage of the language.

The symbols used in this article are introduced in the following vocabularies: FINSEQ, ZF_LANG, FUNC_REL, FUNC, BOOLE, REAL_1, and NAT_1. The terminology and notation used in this article have been introduced in the following articles: TARSKI, BOOLE, FUNCT_1, REAL_1, NAT_1, and FINSEQ_1.

reserve k, l, m, n **for** Nat, X, Y, Z **for** set, D, D1, D2 **for** DOMAIN, a, b, c, d **for** Any.

reserve p, q, r, p′, q′ **for** FinSequence **of** NAT.

Definition

func VAR → SUBDOMAIN **of** NAT **means it** = {k: 5 ⩽ k}.

Theorem ZF_LANG:1. VAR = {k: 5 ⩽ k}.

Definition

mode Variable → Element **of** VAR **means not contradiction**.

[1] Supported by RPBP.III-24.C1.

Theorem ZF_LANG:2. a **is** Variable **iff** a **is** Element **of** VAR.

Definition

let n.

func ξn → Variable **means** **it** = 5+n.

Theorem ZF_LANG:3. ξn = 5+n.

reserve x, y, z, t, s **for** Variable.

Definition

let x.

redefine

func ⟨x⟩ → FinSequence **of** NAT.

Definition

let x, y.

func x‘=’y → FinSequence **of** NAT **means** **it** = ⟨0⟩⌢⟨x⟩⌢⟨y⟩.

func x‘∈’y → FinSequence **of** NAT **means** **it** = ⟨1⟩⌢⟨x⟩⌢⟨y⟩.

Theorem ZF_LANG:4. x‘=’y = ⟨0⟩⌢⟨x⟩⌢⟨y⟩.

Theorem ZF_LANG:5. x‘∈’y = ⟨1⟩⌢⟨x⟩⌢⟨y⟩.

Theorem ZF_LANG:6. x‘=’y = z‘=’t **implies** x = z & y = t.

Theorem ZF_LANG:7. x‘∈’y = z‘∈’t **implies** x = z & y = t.

Definition

let p.

func ¬p → FinSequence **of** NAT **means** **it** = ⟨2⟩⌢p.

let q.

func p∧q → FinSequence **of** NAT **means** **it** = ⟨3⟩⌢p⌢q.

Theorem ZF_LANG:8. ¬p = ⟨2⟩⌢p.

Theorem ZF_LANG:9. p∧q = ⟨3⟩⌢p⌢q.

Theorem ZF_LANG:10. ¬p = ¬q **implies** p = q.

Definition

let x, p.

func ∀(x, p) → FinSequence **of** NAT **means** **it** = ⟨4⟩⌢⟨x⟩⌢p.

Theorem ZF_LANG:11. ∀(x, p) = ⟨4⟩⌢⟨x⟩⌢p.

Theorem ZF_LANG:12. ∀(x, p) = ∀(y, q) **implies** x = y & p = q.

Definition

func WFF → DOMAIN **means** (**for** a **st** a ∈ **it** **holds** a **is** FinSequence **of** NAT) & (**for** x, y **holds** x‘=’y ∈ **it** & x‘∈’y ∈ **it**) & (**for** p **st** p ∈ **it** **holds** ¬p ∈ **it**) & (**for** p, q **st** p ∈ **it** & q ∈ **it** **holds** p∧q ∈ **it**) & (**for** x, p **st** p ∈ **it** **holds** ∀(x, p) ∈ **it**) & **for** D **st** (**for** a **st** a ∈ D **holds** a **is** FinSequence **of** NAT) & (**for** x, y **holds** x‘=’y ∈ D & x‘∈’y

∈ D) & (**for** p **st** p ∈ D **holds** ¬p ∈ D) & (**for** p, q **st** p ∈ D & q ∈ D **holds** p∧q ∈ D) & (**for** x, p **st** p ∈ D **holds** ∀(x, p) ∈ D) **holds** **it** ⊆ D.

Theorem ZF_LANG:13. (**for** a **st** a ∈ WFF **holds** a **is** FinSequence **of** NAT) & (**for** x, y **holds** x‘=’y ∈ WFF & x‘∈’y ∈ WFF) & (**for** p **st** p ∈ WFF **holds** ¬p ∈ WFF) & (**for** p, q **st** p ∈ WFF & q ∈ WFF **holds** p∧q ∈ WFF) & (**for** x, p **st** p ∈ WFF **holds** ∀(x, p) ∈ WFF) & **for** D **st** (**for** a **st** a ∈ D **holds** a **is** FinSequence **of** NAT) & (**for** x, y **holds** x‘=’y ∈ D & x‘∈’y ∈ D) & (**for** p **st** p ∈ D **holds** ¬p ∈ D) & (**for** p, q **st** p ∈ D & q ∈ D **holds** p∧q ∈ D) & (**for** x, p **st** p ∈ D **holds** ∀(x, p) ∈ D) **holds** WFF ⊆ D.

Definition

mode ZF-formula → FinSequence **of** NAT **means** **it** **is** Element **of** WFF.

Theorem ZF_LANG:14. a **is** ZF-formula **iff** a ∈ WFF.

Theorem ZF_LANG:15. a **is** ZF-formula **iff** a **is** Element **of** WFF.

reserve F, F1, G, G1, H, H1 **for** ZF-formula.

Definition

let x, y.

redefine

func x‘=’y → ZF-formula.

func x‘∈’y → ZF-formula.

Definition

let H.

redefine

func ¬H → ZF-formula.

let G.

func H∧G → ZF-formula.

Definition

let x, H.

redefine

func ∀(x, H) → ZF-formula.

Definition

let H.

pred H is equality **means** **ex** x, y **st** H = x‘=’y.

pred H is membership **means** **ex** x, y **st** H = x‘∈’y.

pred H is negative **means** **ex** H1 **st** H = ¬H1.

pred H is conjunctive **means** **ex** F, G **st** H = F∧G.

pred H is universal **means** **ex** x, H1 **st** H = ∀(x, H1).

Theorem ZF_LANG:16. (H is equality **iff** **ex** x, y **st** H = x'='y) & (H is membership **iff** **ex** x, y **st** H = x'∈'y) & (H is negative **iff** **ex** H1 **st** H = ¬H1) & (H is conjunctive **iff** **ex** F, G **st** H = F∧G) & (H is universal **iff** **ex** x, H1 **st** H = ∀(x, H1)).

Definition

let H.

pred H is atomic **means** H is equality **or** H is membership.

Theorem ZF_LANG:17. H is atomic **iff** H is equality **or** H is membership.

Definition

let F, G.

func F∨G → ZF-formula **means** **it** = ¬(¬F∧¬G).

func F⇒G → ZF-formula **means** **it** = ¬(F∧¬G).

Theorem ZF_LANG:18. F∨G = ¬(¬F∧¬G).

Theorem ZF_LANG:19. F⇒G = ¬(F∧¬G).

Definition

let F, G.

func F⇔G → ZF-formula **means** **it** = (F⇒G)∧(G⇒F).

Theorem ZF_LANG:20. F⇔G = (F⇒G)∧(G⇒F).

Definition

let x, H.

func ∃(x, H) → ZF-formula **means** **it** = ¬∀(x, ¬H).

Theorem ZF_LANG:21. ∃(x, H) = ¬∀(x, ¬H).

Definition

let H.

pred H is disjunctive **means** **ex** F, G **st** H = F∨G.

pred H is conditional **means** **ex** F, G **st** H = F⇒G.

pred H is biconditional **means** **ex** F, G **st** H = F⇔G.

pred H is existential **means** **ex** x, H1 **st** H = ∃(x, H1).

Theorem ZF_LANG:22. (H is disjunctive **iff** **ex** F, G **st** H = F∨G) & (H is conditional **iff** **ex** F, G **st** H = F⇒G) & (H is biconditional **iff** **ex** F, G **st** H = F⇔G) & (H is existential **iff** **ex** x, H1 **st** H = ∃(x, H1)).

Definition

let x, y, H.

func ∀(x, y, H) → ZF-formula **means** **it** = ∀(x, ∀(y, H)).

func ∃(x, y, H) → ZF-formula **means** **it** = ∃(x, ∃(y, H)).

Theorem ZF_LANG:23. ∀(x, y, H) = ∀(x, ∀(y, H)) & ∃(x, y, H) = ∃(x, ∃(y, H)).

Definition

let x, y, z, H.

func ∀(x, y, z, H) → ZF-formula **means it** = ∀(x, ∀(y, z, H)).

func ∃(x, y, z, H) → ZF-formula **means it** = ∃(x, ∃(y, z, H)).

Theorem ZF_LANG:24. ∀(x, y, z, H) = ∀(x, ∀(y, z, H)) & ∃(x, y, z, H) = ∃(x, ∃(y, z, H)).

Theorem ZF_LANG:25. H is equality **or** H is membership **or** H is negative **or** H is conjunctive **or** H is universal.

Theorem ZF_LANG:26. H is atomic **or** H is negative **or** H is conjunctive **or** H is universal.

Theorem ZF_LANG:27. H is atomic **implies** len H = 3.

Theorem ZF_LANG:28. H is atomic **or ex** H1 **st** len H1+1 ⩽ len H.

Theorem ZF_LANG:29. 3 ⩽ len H.

Theorem ZF_LANG:30. len H = 3 **implies** H is atomic.

reserve p, q, r **for** ZF-formula.

Theorem ZF_LANG:31. **for** x, y **holds** (x‘=’y).1 = 0 & (x‘∈’y).1 = 1.

Theorem ZF_LANG:32. **for** H **holds** (¬H).1 = 2.

Theorem ZF_LANG:33. **for** F, G **holds** (F∧G).1 = 3.

Theorem ZF_LANG:34. **for** x, H **holds** ∀(x, H).1 = 4.

Theorem ZF_LANG:35. H is equality **implies** H.1 = 0.

Theorem ZF_LANG:36. H is membership **implies** H.1 = 1.

Theorem ZF_LANG:37. H is negative **implies** H.1 = 2.

Theorem ZF_LANG:38. H is conjunctive **implies** H.1 = 3.

Theorem ZF_LANG:39. H is universal **implies** H.1 = 4.

Theorem ZF_LANG:40. H is equality & H.1 = 0 **or** H is membership & H.1 = 1 **or** H is negative & H.1 = 2 **or** H is conjunctive & H.1 = 3 **or** H is universal & H.1 = 4.

Theorem ZF_LANG:41. H.1 = 0 **implies** H is equality.

Theorem ZF_LANG:42. H.1 = 1 **implies** H is membership.

Theorem ZF_LANG:43. H.1 = 2 **implies** H is negative.

Theorem ZF_LANG:44. H.1 = 3 **implies** H is conjunctive.

Theorem ZF_LANG:45. H.1 = 4 **implies** H is universal.

reserve sq, sq′ **for** FinSequence.

Theorem ZF_LANG:46. H = F⌒sq **implies** H = F.

Theorem ZF_LANG:47. H∧G = H1∧G1 **implies** H = H1 & G = G1.

Theorem ZF_LANG:48. F∨G = F1∨G1 **implies** F = F1 & G = G1.

Theorem ZF_LANG:49. F⇒G = F1⇒G1 **implies** F = F1 & G = G1.

Theorem ZF_LANG:50. F⇔G = F1⇔G1 **implies** F = F1 & G = G1.

Theorem ZF_LANG:51. ∃(x, H) = ∃(y, G) **implies** x = y & H = G.

Definition

let H.

assume H is atomic.

func Var_1H → Variable **means** **it** = H.2.

func Var_2H → Variable **means** **it** = H.3.

Theorem ZF_LANG:52. H is atomic **implies** Var_1H = H.2 & Var_2H = H.3.

Theorem ZF_LANG:53. H is equality **implies** H = (Var_1H)'='Var_2H.

Theorem ZF_LANG:54. H is membership **implies** H = (Var_1H)'∈'Var_2H.

Definition

let H.

assume H is negative.

func the argument of H → ZF-formula **means** ¬**it** = H.

Theorem ZF_LANG:55. H is negative **implies** H = ¬the argument of H.

Definition

let H.

assume H is conjunctive **or** H is disjunctive.

func the left argument of H → ZF-formula **means** **ex** H1 **st** **it**∧H1 = H **if** H is conjunctive **otherwise** **ex** H1 **st** **it**∨H1 = H.

func the right argument of H → ZF-formula **means** **ex** H1 **st** H1∧**it** = H **if** H is conjunctive **otherwise** **ex** H1 **st** H1∨**it** = H.

Theorem ZF_LANG:56. H is conjunctive **implies** (F = the left argument of H **iff** **ex** G **st** F∧G = H) & (F = the right argument of H **iff** **ex** G **st** G∧F = H).

Theorem ZF_LANG:57. H is disjunctive **implies** (F = the left argument of H **iff** **ex** G **st** F∨G = H) & (F = the right argument of H **iff** **ex** G **st** G∨F = H).

Theorem ZF_LANG:58. H is conjunctive **implies** H = (the left argument of H)∧the right argument of H.

Theorem ZF_LANG:59. H is disjunctive **implies** H = (the left argument of H)∨the right argument of H.

Definition

let H.

assume H is universal **or** H is existential.

func bound in H → Variable **means** **ex** H1 **st** ∀(**it**, H1) = H **if** H is universal **otherwise** **ex** H1 **st** ∃(**it**, H1) = H.

func the scope of H → ZF-formula **means** **ex** x **st** ∀(x, **it**) = H **if** H is universal **otherwise** **ex** x **st** ∃(x, **it**) = H.

Theorem ZF_LANG:60. H is universal **implies** (x = bound in H **iff** **ex** H1 **st** ∀(x, H1) = H) & (H1 = the scope of H **iff** **ex** x **st** ∀(x, H1) = H).

Theorem ZF_LANG:61. H is existential **implies** (x = bound in H **iff** **ex** H1 **st** ∃(x, H1) = H) & (H1 = the scope of H **iff** **ex** x **st** ∃(x, H1) = H).

Theorem ZF_LANG:62. H is universal **implies** H = ∀(bound in H, the scope of H).

Theorem ZF_LANG:63. H is existential **implies** H = ∃(bound in H, the scope of H).

Definition

let H.

assume H is conditional.

func the antecedent of H → ZF-formula **means** **ex** H1 **st** H = **it**⇒H1.

func the consequent of H → ZF-formula **means** **ex** H1 **st** H = H1⇒**it**.

Theorem ZF_LANG:64. H is conditional **implies** (F = the antecedent of H **iff** **ex** G **st** H = F⇒G) & (F = the consequent of H **iff** **ex** G **st** H = G⇒F).

Theorem ZF_LANG:65. H is conditional **implies** H = (the antecedent of H)⇒the consequent of H.

Definition

let H.

assume H is biconditional.

func the left side of H → ZF-formula **means** **ex** H1 **st** H = **it**⇔H1.

func the right side of H → ZF-formula **means** **ex** H1 **st** H = H1⇔**it**.

Theorem ZF_LANG:66. H is biconditional **implies** (F = the left side of H **iff** **ex** G **st** H = F⇔G) & (F = the right side of H **iff** **ex** G **st** H = G⇔F).

Theorem ZF_LANG:67. H is biconditional **implies** H = (the left side of H)⇔the right side of H.

Definition

let H, F.

pred H is immediate constituent of F **means** F = ¬H **or** (**ex** H1 **st** F = H∧H1 **or** F = H1∧H) **or** **ex** x **st** F = ∀(x, H).

Theorem ZF_LANG:68. H is immediate constituent of F **iff** F = ¬H **or** (**ex** H1 **st** F = H∧H1 **or** F = H1∧H) **or** **ex** x **st** F = ∀(x, H).

Theorem ZF_LANG:69. **not** H is immediate constituent of x‘=’y.

Theorem ZF_LANG:70. **not** H is immediate constituent of x‘∈’y.

Theorem ZF_LANG:71. F is immediate constituent of ¬H **iff** F = H.

Theorem ZF_LANG:72. F is immediate constituent of G∧H **iff** F = G **or** F = H.

Theorem ZF_LANG:73. F is immediate constituent of ∀(x, H) **iff** F = H.

Theorem ZF_LANG:74. H is atomic **implies not** F is immediate constituent of H.

Theorem ZF_LANG:75. H is negative **implies** (F is immediate constituent of H **iff** F = the argument of H).

Theorem ZF_LANG:76. H is conjunctive **implies** (F is immediate constituent of H **iff** F = the left argument of H **or** F = the right argument of H).

Theorem ZF_LANG:77. H is universal **implies** (F is immediate constituent of H **iff** F = the scope of H).

reserve L, L′ **for** FinSequence, f **for** Function.

Definition

let H, F.

pred H is subformula of F **means ex** n, L **st** 1 ⩽ n & len L = n & L.1 = H & L.n = F & **for** k **st** 1 ⩽ k & k < n **ex** H1, F1 **st** L.k = H1 & L.(k+1) = F1 & H1 is immediate constituent of F1.

Theorem ZF_LANG:78. H is subformula of F **iff ex** n, L **st** 1 ⩽ n & len L = n & L.1 = H & L.n = F & **for** k **st** 1 ⩽ k & k < n **ex** H1, F1 **st** L.k = H1 & L.(k+1) = F1 & H1 is immediate constituent of F1.

Theorem ZF_LANG:79. H is subformula of H.

Definition

let H, F.

pred H is proper subformula of F **means** H is subformula of F & H ≠ F.

Theorem ZF_LANG:80. H is proper subformula of F **iff** H is subformula of F & H ≠ F.

Theorem ZF_LANG:81. H is immediate constituent of F **implies** len H < len F.

Theorem ZF_LANG:82. H is immediate constituent of F **implies** H is proper subformula of F.

Theorem ZF_LANG:83. H is proper subformula of F **implies** len H < len F.

Theorem ZF_LANG:84. H is proper subformula of F **implies ex** G **st** G is immediate constituent of F.

reserve j, j1, j2 **for** Nat.

Theorem ZF_LANG:85. F is proper subformula of G & G is proper subformula of H **implies** F is proper subformula of H.

Theorem ZF_LANG:86. F is subformula of G & G is subformula of H **implies** F is subformula of H.

Theorem ZF_LANG:87. G is subformula of H & H is subformula of G **implies** G = H.

Theorem ZF_LANG:88. **not** F is proper subformula of x‘=’y.

Theorem ZF_LANG:89. **not** F is proper subformula of x‘∈’y.

Theorem ZF_LANG:90. F is proper subformula of ¬H **implies** F is subformula of H.

Theorem ZF_LANG:91. F is proper subformula of G∧H **implies** F is subformula of G **or** F is subformula of H.

Theorem ZF_LANG:92. F is proper subformula of ∀(x, H) **implies** F is subformula of H.

Theorem ZF_LANG:93. H is atomic **implies not** F is proper subformula of H.

Theorem ZF_LANG:94. H is negative **implies** the argument of H is proper subformula of H.

Theorem ZF_LANG:95. H is conjunctive **implies** the left argument of H is proper subformula of H & the right argument of H is proper subformula of H.

Theorem ZF_LANG:96. H is universal **implies** the scope of H is proper subformula of H.

Theorem ZF_LANG:97. H is subformula of x‘=’y **iff** H = x‘=’y.

Theorem ZF_LANG:98. H is subformula of x‘∈’y **iff** H = x‘∈’y.

Definition

let H.

func Subformulae H → set **means** a ∈ **it** **iff** **ex** F **st** F = a & F is subformula of H.

Theorem ZF_LANG:99. a ∈ Subformulae H **iff** **ex** F **st** F = a & F is subformula of H.

Theorem ZF_LANG:100. G ∈ Subformulae H **implies** G is subformula of H.

Theorem ZF_LANG:101. F is subformula of H **implies** Subformulae F ⊆ Subformulae H.

Theorem ZF_LANG:102. Subformulae x‘=’y = {x‘=’y}.

Theorem ZF_LANG:103. Subformulae x‘∈’y = {x‘∈’y}.

Theorem ZF_LANG:104. Subformulae ¬H = Subformulae H∪{¬H}.

Theorem ZF_LANG:105. Subformulae (H∧F) = Subformulae H∪Subformulae F∪{H∧F}.

Theorem ZF_LANG:106. Subformulae ∀(x, H) = Subformulae H∪{∀(x, H)}.

Theorem ZF_LANG:107. H is atomic **iff** Subformulae H = {H}.

Theorem ZF_LANG:108. H is negative **implies** Subformulae H = Subformulae the argument of H∪{H}.

Theorem ZF_LANG:109. H is conjunctive **implies** Subformulae H = Subformulae the left argument of H∪Subformulae the right argument of H∪{H}.

Theorem ZF_LANG:110. H is universal **implies** Subformulae H = Subformulae the scope of H∪{H}.

Theorem ZF_LANG:111. (H is immediate constituent of G **or** H is proper subformula of G **or** H is subformula of G) & G ∈ Subformulae F **implies** H ∈ Subformulae F.

scheme ZF_Ind{P[ZF-formula]}: **for** H **holds** P[H] **provided** A: **for** H **st** H is atomic **holds** P[H] **and** B: **for** H **st** H is negative & P[the argument of H] **holds** P[H] **and** C: **for**

H **st** H is conjunctive & P[the left argument of H] & P[the right argument of H] **holds** P[H] **and** D: **for** H **st** H is universal & P[the scope of H] **holds** P[H].

scheme ZF_CompInd{P[ZF-formula]}: **for** H **holds** P[H] **provided** A: **for** H **st** **for** F **st** F is proper subformula of H **holds** P[F] **holds** P[H].

Chapter 33

ZF_MODEL

Models and Satisfiability

Defining by Structural Induction and Free Variables in ZF-formulae

by

Grzegorz Bancerek [1]

Warsaw University (Białystok)

Summary. The article includes schemes of defining by structural induction, and definitions and theorems related to: the set of variables which have free occurrences in a ZF-formula, the set of all valuations of variables in a model, the set of all valuations which satisfy a ZF-formula in a model, the satisfiability of a ZF-formula in a model by a valuation, the validity of a ZF-formula in a model, the axioms of ZF-language, the model of the ZF set theory.

The symbols used in this article are introduced in the following vocabularies: FINSEQ, ZF_LANG, ZF_SAT, ZF_AXIOM, ORDINAL, FUNC_REL, FUNC, FAM_OP, BOOLE, REAL_1, and NAT_1. The terminology and notation used in this article have been introduced in the following articles: TARSKI, BOOLE, FUNCT_1, REAL_1, NAT_1, FINSEQ_1, ZF_LANG, FUNCT_2, ENUMSET1, and ORDINAL1.

reserve F, G, H, H′ **for** ZF-formula, f, g, h **for** Function, x, y, z, t **for** Variable, a, b, c, d **for** Any, A, X, Y, Z **for** set, D **for** DOMAIN.

scheme ZFsch_ex{F1(Variable, Variable) → Any, F2(Variable, Variable) → Any, F3(Any) → Any, F4(Any, Any) → Any, F5(Variable, Any) → Any, H() → ZF-formula}: **ex** a, A **st**

[1] Supported by RPBP.III-24.C1.

(**for** x, y **holds** [x‘=’y, F1(x, y)] ∈ A & [x‘∈’y, F2(x, y)] ∈ A) & [H(), a] ∈ A & **for** H, a **st** [H, a] ∈ A **holds** (H is equality **implies** a = F1(Var_1H, Var_2H)) & (H is membership **implies** a = F2(Var_1H, Var_2H)) & (H is negative **implies** **ex** b **st** a = F3(b) & [the argument of H, b] ∈ A) & (H is conjunctive **implies** **ex** b, c **st** (a = F4(b, c) & [the left argument of H, b] ∈ A) & [the right argument of H, c] ∈ A) & (H is universal **implies** **ex** b, x **st** x = bound in H & a = F5(x, b) & [the scope of H, b] ∈ A).

scheme ZFsch_uniq{F1(Variable, Variable) → Any, F2(Variable, Variable) → Any, F3(Any) → Any, F4(Any, Any) → Any, F5(Variable, Any) → Any, H() → ZF-formula, a() → Any, b() → Any}: a() = b() **provided** A: **ex** A **st** (**for** x, y **holds** [x‘=’y, F1(x, y)] ∈ A & [x‘∈’y, F2(x, y)] ∈ A) & [H(), a()] ∈ A & **for** H, a **st** [H, a] ∈ A **holds** (H is equality **implies** a = F1(Var_1H, Var_2H)) & (H is membership **implies** a = F2(Var_1H, Var_2H)) & (H is negative **implies** **ex** b **st** a = F3(b) & [the argument of H, b] ∈ A) & (H is conjunctive **implies** **ex** b, c **st** a = F4(b, c) & [the left argument of H, b] ∈ A & [the right argument of H, c] ∈ A) & (H is universal **implies** **ex** b, x **st** x = bound in H & a = F5(x, b) & [the scope of H, b] ∈ A) **and** B: **ex** A **st** (**for** x, y **holds** [x‘=’y, F1(x, y)] ∈ A & [x‘∈’y, F2(x, y)] ∈ A) & [H(), b()] ∈ A & **for** H, a **st** [H, a] ∈ A **holds** (H is equality **implies** a = F1(Var_1H, Var_2H)) & (H is membership **implies** a = F2(Var_1H, Var_2H)) & (H is negative **implies** **ex** b **st** a = F3(b) & [the argument of H, b] ∈ A) & (H is conjunctive **implies** **ex** b, c **st** a = F4(b, c) & [the left argument of H, b] ∈ A & [the right argument of H, c] ∈ A) & (H is universal **implies** **ex** b, x **st** x = bound in H & a = F5(x, b) & [the scope of H, b] ∈ A).

scheme ZFsch_result{F1(Variable, Variable) → Any, F2(Variable, Variable) → Any, F3(Any) → Any, F4(Any, Any) → Any, F5(Variable, Any) → Any, H() → ZF-formula, f(ZF-formula) → Any}: (H() is equality **implies** f(H()) = F1(Var_1H(), Var_2H())) & (H() is membership **implies** f(H()) = F2(Var_1H(), Var_2H())) & (H() is negative **implies** f(H()) = F3(f(the argument of H()))) & (H() is conjunctive **implies** **for** a, b **st** a = f(the left argument of H()) & b = f(the right argument of H()) **holds** f(H()) = F4(a, b)) & (H() is universal **implies** f(H()) = F5(bound in H(), f(the scope of H()))) **provided** A: **for** H′, a **holds** a = f(H′) **iff** **ex** A **st** (**for** x, y **holds** [x‘=’y, F1(x, y)] ∈ A & [x‘∈’y, F2(x, y)] ∈ A) & [H′, a] ∈ A & **for** H, a **st** [H, a] ∈ A **holds** (H is equality **implies** a = F1(Var_1H, Var_2H)) & (H is membership **implies** a = F2(Var_1H, Var_2H)) & (H is negative **implies** **ex** b **st** a = F3(b) & [the argument of H, b] ∈ A) & (H is conjunctive **implies** **ex** b, c **st** a = F4(b, c) & [the left argument of H, b] ∈ A & [the right argument of H, c] ∈ A) & (H is universal **implies** **ex** b, x **st** x = bound in H & a = F5(x, b) & [the scope of H, b] ∈ A).

scheme ZFsch_property{F1(Variable, Variable) → Any, F2(Variable, Variable) → Any, F3(Any) → Any, F4(Any, Any) → Any, F5(Variable, Any) → Any, H() → ZF-formula, f(ZF-formula) → Any, P[Any]}: P[f(H())] **provided** A: **for** H′, a **holds** a = f(H′) **iff** **ex** A **st** (**for** x, y **holds** [x‘=’y, F1(x, y)] ∈ A & [x‘∈’y, F2(x, y)] ∈ A) & [H′, a] ∈ A & **for** H, a **st** [H, a] ∈ A **holds** (H is equality **implies** a = F1(Var_1H, Var_2H)) & (H is membership **implies** a = F2(Var_1H, Var_2H)) & (H is negative **implies** **ex** b **st** a = F3(b) & [the argument of H, b] ∈ A) & (H is conjunctive **implies** **ex** b, c **st** a = F4(b, c) & [the left argument of H, b] ∈ A & [the right argument of H, c] ∈ A) & (H is universal **implies** **ex** b, x **st** x = bound in H & a = F5(x, b) & [the scope of H, b] ∈ A) **and** B: **for** x, y **holds** P[F1(x, y)] & P[F2(x, y)] **and** C: **for** a **st** P[a] **holds** P[F3(a)] **and** D: **for** a, b **st** P[a] & P[b] **holds** P[F4(a, b)] **and** E: **for** a, x **st** P[a] **holds** P[F5(x, a)].

Definition

let H.

func Free H → Any **means** **ex** A **st** (**for** x, y **holds** [x'='y, {x, y}] ∈ A & [x'∈'y, {x, y}] ∈ A) & [H, **it**] ∈ A & **for** H′, a **st** [H′, a] ∈ A **holds** (H′ is equality **implies** a = {Var_1H′, Var_2H′}) & (H′ is membership **implies** a = {Var_1H′, Var_2H′}) & (H′ is negative **implies** **ex** b **st** a = b & [the argument of H′, b] ∈ A) & (H′ is conjunctive **implies** **ex** b, c **st** a = ⋃{b, c} & [the left argument of H′, b] ∈ A & [the right argument of H′, c] ∈ A) & (H′ is universal **implies** **ex** b, x **st** x = bound in H′ & a = (⋃{b})∖{x} & [the scope of H′, b] ∈ A).

Definition

let H.

redefine

func Free H → **set** **of** Variable.

Theorem ZF_MODEL:1. **for** H **holds** (H is equality **implies** Free H = {Var_1H, Var_2H}) & (H is membership **implies** Free H = {Var_1H, Var_2H}) & (H is negative **implies** Free H = Free the argument of H) & (H is conjunctive **implies** Free H = Free the left argument of H∪Free the right argument of H) & (H is universal **implies** Free H = (Free the scope of H)∖{bound in H}).

Definition

let D **be** SET DOMAIN.

func VAL D → DOMAIN **means** a ∈ **it** **iff** a **is** Function **of** VAR, D.

Definition

let D1 **be** SET DOMAIN, f **be** Function **of** VAR, D1.

let x.

redefine

func f.x → Element **of** D1.

reserve E **for** SET DOMAIN, f, g, h **for** (Function **of** VAR, E), v1, v2, v3, v4, v5, u1, u2, u3, u4, u5 **for** (Element **of** VAL E), S, T **for** Subset **of** ⟦WFF, VAL E⟧.

Definition

let H, E.

func St (H, E) → Any **means** **ex** A **st** (**for** x, y **holds** [x'='y, {v1: **for** f **st** f = v1 **holds** f.x = f.y}] ∈ A & [x'∈'y, {v2: **for** f **st** f = v2 **holds** f.x ∈ f.y}] ∈ A) & [H, **it**] ∈ A & **for** H′, a **st** [H′, a] ∈ A **holds** (H′ is equality **implies** a = {v3: **for** f **st** f = v3 **holds** f.(Var_1H′) = f.(Var_2H′)}) & (H′ is membership **implies** a = {v4: **for** f **st** f = v4 **holds** f.(Var_1H′) ∈ f.(Var_2H′)}) & (H′ is negative **implies** **ex** b **st** a = (VAL E)∖⋃{b} & [the argument of H′, b] ∈ A) & (H′ is conjunctive **implies ex** b, c **st** a = (⋃{b})∩⋃{c} & [the left argument of H′, b] ∈ A & [the right argument of H′, c] ∈ A) & (H′ is universal **implies ex** b, x **st** x = bound in H′ & a = {v5: **for** X, f **st** X = b & f = v5 **holds** f ∈ X & **for** g **st for** y **st** g.y ≠ f.y **holds** x = y **holds** g ∈ X} & [the scope of H′, b] ∈ A).

Definition

let H, E.

redefine

func St (H, E) → Subset **of** VAL E.

Theorem ZF_MODEL:2. **for** x, y, f **holds** f.x = f.y **iff** f ∈ St (x‘=’y, E).

Theorem ZF_MODEL:3. **for** x, y, f **holds** f.x ∈ f.y **iff** f ∈ St (x‘∈’y, E).

Theorem ZF_MODEL:4. **for** H, f **holds not** f ∈ St (H, E) **iff** f ∈ St (¬H, E).

Theorem ZF_MODEL:5. **for** H, H′, f **holds** f ∈ St (H, E) & f ∈ St (H′, E) **iff** f ∈ St (H∧H′, E).

Theorem ZF_MODEL:6. **for** x, H, f **holds** (f ∈ St (H, E) & **for** g **st for** y **st** g.y ≠ f.y **holds** x = y **holds** g ∈ St (H, E)) **iff** f ∈ St (∀(x, H), E).

Theorem ZF_MODEL:7. H is equality **implies for** f **holds** f.(Var_1H) = f.(Var_2H) **iff** f ∈ St (H, E).

Theorem ZF_MODEL:8. H is membership **implies for** f **holds** f.(Var_1H) ∈ f.(Var_2H) **iff** f ∈ St (H, E).

Theorem ZF_MODEL:9. H is negative **implies for** f **holds not** f ∈ St (the argument of H, E) **iff** f ∈ St (H, E).

Theorem ZF_MODEL:10. H is conjunctive **implies for** f **holds** f ∈ St (the left argument of H, E) & f ∈ St (the right argument of H, E) **iff** f ∈ St (H, E).

Theorem ZF_MODEL:11. H is universal **implies for** f **holds** (f ∈ St (the scope of H, E) & **for** g **st for** y **st** g.y ≠ f.y **holds** bound in H = y **holds** g ∈ St (the scope of H, E)) **iff** f ∈ St (H, E).

Definition

let D **be** SET DOMAIN.

let f **be** Function **of** VAR, D.

let H.

pred D, f ⊨ H **means** f ∈ St (H, D).

Theorem ZF_MODEL:12. **for** E, f, x, y **holds** E, f ⊨ x‘=’y **iff** f.x = f.y.

Theorem ZF_MODEL:13. **for** E, f, x, y **holds** E, f ⊨ x‘∈’y **iff** f.x ∈ f.y.

Theorem ZF_MODEL:14. **for** E, f, H **holds** E, f ⊨ H **iff not** E, f ⊨ ¬H.

Theorem ZF_MODEL:15. **for** E, f, H, H′ **holds** E, f ⊨ H∧H′ **iff** E, f ⊨ H & E, f ⊨ H′.

Theorem ZF_MODEL:16. **for** E, f, H, x **holds** E, f $\models$ $\forall$(x, H) **iff** **for** g **st** **for** y **st** g.y $\neq$ f.y **holds** x = y **holds** E, g $\models$ H.

Theorem ZF_MODEL:17. **for** E, f, H, H$'$ **holds** E, f $\models$ H$\vee$H$'$ **iff** E, f $\models$ H **or** E, f $\models$ H$'$.

Theorem ZF_MODEL:18. **for** E, f, H, H$'$ **holds** E, f $\models$ H$\Rightarrow$H$'$ **iff** (E, f $\models$ H **implies** E, f $\models$ H$'$).

Theorem ZF_MODEL:19. **for** E, f, H, H$'$ **holds** E, f $\models$ H$\Leftrightarrow$H$'$ **iff** (E, f $\models$ H **iff** E, f $\models$ H$'$).

Theorem ZF_MODEL:20. **for** E, f, H, x **holds** E, f $\models$ $\exists$(x, H) **iff** **ex** g **st** (**for** y **st** g.y $\neq$ f.y **holds** x = y) & E, g $\models$ H.

Theorem ZF_MODEL:21. **for** E, f, x **for** e **being** Element **of** E **ex** g **st** g.x = e & **for** z **st** z $\neq$ x **holds** g.z = f.z.

Theorem ZF_MODEL:22. E, f $\models$ $\forall$(x, y, H) **iff** **for** g **st** **for** z **st** g.z $\neq$ f.z **holds** x = z **or** y = z **holds** E, g $\models$ H.

Theorem ZF_MODEL:23. E, f $\models$ $\exists$(x, y, H) **iff** **ex** g **st** (**for** z **st** g.z $\neq$ f.z **holds** x = z **or** y = z) & E, g $\models$ H.

Definition

let E, H.

pred E $\models$ H **means** **for** f **holds** E, f $\models$ H.

Theorem ZF_MODEL:24. E $\models$ H **iff** **for** f **holds** E, f $\models$ H.

Theorem ZF_MODEL:25. E $\models$ $\forall$(x, H) **iff** E $\models$ H.

Definition

func the axiom of extensionality $\rightarrow$ ZF-formula **means** **it** = $\forall$(ξ0, ξ1, $\forall$(ξ2, ξ2'$\in$' ξ0$\Leftrightarrow$$\xi$2'$\in$'$\xi$1)$\Rightarrow$$\xi$0'='$\xi$1).

func the axiom of pairs $\rightarrow$ ZF-formula **means** **it** = $\forall$(ξ0, ξ1, $\exists$(ξ2, $\forall$(ξ3, ξ3'$\in$' ξ2$\Leftrightarrow$(ξ3'='ξ0$\vee$$\xi$3'='$\xi$1)))).

func the axiom of unions $\rightarrow$ ZF-formula **means** **it** = $\forall$(ξ0, $\exists$(ξ1, $\forall$(ξ2, ξ2'$\in$'ξ1$\Leftrightarrow$ $\exists$(ξ3, ξ2'$\in$'ξ3$\wedge$$\xi$3'$\in$'$\xi$0)))).

func the axiom of infinity $\rightarrow$ ZF-formula **means** **it** = $\exists$(ξ0, ξ1, ξ1'$\in$'ξ0$\wedge$$\forall$($\xi$2, ξ2'$\in$'ξ0$\Rightarrow$$\exists$($\xi$3, ξ3'$\in$'ξ0$\wedge$$\neg$$\xi$3'='$\xi$2$\wedge$$\forall$($\xi$4, ξ4'$\in$'ξ2$\Rightarrow$$\xi$4'$\in$'$\xi$3)))).

func the axiom of power sets $\rightarrow$ ZF-formula **means** **it** = $\forall$(ξ0, $\exists$(ξ1, $\forall$(ξ2, ξ2'$\in$' ξ1$\Leftrightarrow$$\forall$($\xi$3, ξ3'$\in$'ξ2$\Rightarrow$$\xi$3'$\in$'$\xi$0)))).

Definition

let H **be** ZF-formula.

assume {ξ0, ξ1, ξ2} misses Free H.

func the axiom of substitution for H $\rightarrow$ ZF-formula **means** **it** = $\forall$(ξ3, $\exists$(ξ0, $\forall$(ξ4, H$\Leftrightarrow$$\xi$4'='$\xi$0)))$\Rightarrow$$\forall$($\xi$1, $\exists$(ξ2, $\forall$(ξ4, ξ4'$\in$'ξ2$\Leftrightarrow$$\exists$($\xi$3, ξ3'$\in$'ξ1$\wedge$H)))).

Theorem ZF_MODEL:26. the axiom of extensionality = ∀(ξ0, ξ1, ∀(ξ2, ξ2'∈'ξ0⇔ξ2'∈' ξ1)⇒ξ0'='ξ1).

Theorem ZF_MODEL:27. the axiom of pairs = ∀(ξ0, ξ1, ∃(ξ2, ∀(ξ3, ξ3'∈'ξ2⇔(ξ3'=' ξ0∨ξ3'='ξ1)))).

Theorem ZF_MODEL:28. the axiom of unions = ∀(ξ0, ∃(ξ1, ∀(ξ2, ξ2'∈'ξ1⇔∃(ξ3, ξ2'∈' ξ3∧ξ3'∈'ξ0)))).

Theorem ZF_MODEL:29. the axiom of infinity = ∃(ξ0, ξ1, ξ1'∈'ξ0∧∀(ξ2, ξ2'∈'ξ0⇒ ∃(ξ3, ξ3'∈'ξ0∧¬ξ3'='ξ2∧∀(ξ4, ξ4'∈'ξ2⇒ξ4'∈'ξ3)))).

Theorem ZF_MODEL:30. the axiom of power sets = ∀(ξ0, ∃(ξ1, ∀(ξ2, ξ2'∈'ξ1⇔∀(ξ3, ξ3'∈'ξ2⇒ξ3'∈'ξ0)))).

Theorem ZF_MODEL:31. {ξ0, ξ1, ξ2} misses Free H **implies** the axiom of substitution for H = ∀(ξ3, ∃(ξ0, ∀(ξ4, H⇔ξ4'='ξ0)))⇒∀(ξ1, ∃(ξ2, ∀(ξ4, ξ4'∈'ξ2⇔∃(ξ3, ξ3'∈'ξ1∧H)))).

Definition

let E.

pred E is a model of ZF **means** E is ∈-transitive & E ⊨ the axiom of pairs & E ⊨ the axiom of unions & E ⊨ the axiom of infinity & E ⊨ the axiom of power sets & **for** H **st** {ξ0, ξ1, ξ2} misses Free H **holds** E ⊨ the axiom of substitution for H.

Theorem ZF_MODEL:32. E is a model of ZF **iff** E is ∈-transitive & E ⊨ the axiom of pairs & E ⊨ the axiom of unions & E ⊨ the axiom of infinity & E ⊨ the axiom of power sets & **for** H **st** {ξ0, ξ1, ξ2} misses Free H **holds** E ⊨ the axiom of substitution for H.

Chapter 34

ZF_COLLA

The Contraction Lemma

by

Grzegorz Bancerek [1]

Warsaw University (Białystok)

Summary. The article includes the proof of the contraction lemma which claims that every class in which the axiom of extensionality is valid is isomorphic with a transitive class. In this article the isomorphism (wrt membership relation) of two sets is defined. It is based on *Constructible sets* by A. Mostowski.

The symbols used in this article are introduced in the following vocabularies: FINSEQ, ZF_LANG, ZF_SAT, ZF_AXIOM, COLLAPS, ORDINAL, FUNC_REL, FUNC, BOOLE, FAM_OP, REAL_1, and NAT_1. The terminology and notation used in this article have been introduced in the following articles: TARSKI, BOOLE, FUNCT_1, REAL_1, NAT_1, FINSEQ_1, ZF_LANG, FUNCT_2, ENUMSET1, ORDINAL1, and ZF_MODEL.

reserve X, Y, Z **for** set, v, w, x, y, z **for** Any, E **for** SET DOMAIN, A, B, C **for** Ordinal, L, L1 **for** transfinite sequence, f, f1, f2, g, h **for** Function, d, d1, d2, d′ **for** Element **of** E.

Definition

let E, A.

[1] Supported by RPBP.III-24.C1.

func Mμ(E, A) → set **means** **ex** L **st** **it** = {d: **for** d1 **st** d1 ∈ d **ex** B **st** B ∈ dom L & d1 ∈ $\bigcup${L.B}} & dom L = A & **for** B **st** B ∈ A **holds** L.B = {d1: **for** d **st** d ∈ d1 **ex** C **st** C ∈ dom (L↾B) & d ∈ $\bigcup${L↾B.C}}.

Definition

let f, X, Y.

pred f is ∈-isomorphism of X, Y **means** dom f = X & rng f = Y & f is 1-1 & **for** x, y **st** x ∈ X & y ∈ X **holds** (**ex** Z **st** Z = y & x ∈ Z) **iff** (**ex** Z **st** f.y = Z & f.x ∈ Z).

Definition

let X, Y.

pred X, Y are ∈-isomorphic **means** **ex** f **st** f is ∈-isomorphism of X, Y.

reserve f, g, h **for** (Function **of** VAR, E), u, v, w **for** (Element **of** E), x, y, z **for** Variable, a, b, c **for** Any.

Theorem ZF_COLLA:1. E ⊨ the axiom of extensionality **implies** **for** u, v **st** **for** w **holds** w ∈ u **iff** w ∈ v **holds** u = v.

Theorem ZF_COLLA:2. E ⊨ the axiom of extensionality **implies** **ex** X **st** X is ∈-transitive & E, X are ∈-isomorphic.

Appendix A

Built-in Concepts

This article is written in plain Mizar; no additional vocabularies or signatures are referenced.

Definition

mode Any.

Definition

mode set → Any.

Definition

let x, y **be** Any.

pred x = y.

Definition

let x **be** Any, X **be** set.

pred x ∈ X.

Definition

let X **be** set.

mode Element **of** X.

Definition

mode DOMAIN → set.

Definition

let X **be** DOMAIN.

redefine

mode Element **of** X.

Definition

let X1, X2 **be** set.

func ⟦X1, X2⟧ → set.

let X3 **be** set.

func ⟦X1, X2, X3⟧ → set.

let X4 **be** set.

func ⟦X1, X2, X3, X4⟧ → set.

Definition

let X1, X2 **be** DOMAIN.

redefine

func ⟦X1, X2⟧ → DOMAIN.

let X3 **be** DOMAIN.

func ⟦X1, X2, X3⟧ → DOMAIN.

let X4 **be** DOMAIN.

func ⟦X1, X2, X3, X4⟧ → DOMAIN.

Definition

let X1, X2 **be** DOMAIN.

mode TUPLE **of** X1, X2 → Element **of** ⟦X1, X2⟧ **means not contradiction**.

let X3 **be** DOMAIN.

mode TUPLE **of** X1, X2, X3 → Element **of** ⟦X1, X2, X3⟧ **means not contradiction**.

let X4 **be** DOMAIN.

mode TUPLE **of** X1, X2, X3, X4 → Element **of** ⟦X1, X2, X3, X4⟧ **means not contradiction**.

Definition

let X **be** set.

mode Subset **of** X → set.

func bool X → set.

Definition

mode SET DOMAIN → DOMAIN.

Definition

let D **be** DOMAIN.

redefine

func bool D → SET DOMAIN.

Definition

let D **be** SET DOMAIN.

redefine

mode Element **of** D → set.

Definition
let X **be** DOMAIN.
redefine
mode Subset **of** X → Element **of** bool X **means not contradiction**.

Definition
let X **be** DOMAIN.
mode SUBDOMAIN **of** X → DOMAIN.

Definition
func REAL → DOMAIN.

Definition
func NAT → SUBDOMAIN **of** REAL.

Definition
let x, y **be** Element **of** REAL.
func x+y → Element **of** REAL.
func x·y → Element **of** REAL.
pred x $\leqslant$ y.

Definition
mode Real → Element **of** REAL **means not contradiction**.

Definition
let D **be** DOMAIN, X **be** SUBDOMAIN **of** D.
redefine
mode Element **of** X → Element **of** D.

Definition
let X **be** SUBDOMAIN **of** REAL.
redefine
mode Element **of** X → Real.

Definition
mode Nat → Element **of** NAT **means not contradiction**.

Appendix B

The Grammar of Mizar Abstracts

```
Abstract = "environ" Environment "begin" Text-Proper .
Environment = { Directive } .
Directive =
     "vocabulary" Vocabulary-File-Name  ";" |
     "signature" Signature-File-Name  ";" .
Text-Proper = { Text-Item } .
Text-Item =
      Reservation | Definition-Block |
      Structure-Definition |
      Theorem | Scheme .
Theorem = Compact-Statement .

Reservation =
     "reserve" Reservation-Segment
               { "," Reservation-Segment } ";" .
Reservation-Segment = Reserved-Identifiers-List "for" Type .
Reserved-Identifiers-List = Identifier { "," Identifier } .

Definition-Block =
     "definition" Definitions [ "redefine" Redefinitions ]
     "end" ";".
Definitions = { Definition-Item } .
Redefinitions = { Definition-Item } .
Definition-Item =
     Generalization |
     Assumption |
     Mode-Definition |
     Function-Definition |
     Predicate-Definition .
```

```
Mode-Definition =
     "mode" Mode-Pattern [ Specification ]
        [ "means" Definiens ]  ";" .
Mode-Pattern = Mode-Symbol [ "of" Loci ] .

Function-Definition =
     "func" Function-Pattern [ Specification ]
        [ "means" Definiens ]  ";" .
Function-Pattern =
   [ Function-Loci ] Function-Symbol [ Function-Loci ] |
     Left-Function-Bracket Loci Right-Function-Bracket |
     "{" Loci "}" |
     "[" Loci "]".

Predicate-Definition =
     "pred" Predicate-Pattern [ "means" Definiens ]  ";" .
Predicate-Pattern =
   [ Loci ] Predicate-Symbol [ Loci ] |
     Locus "=" Locus.

Structure-Definition =
     "struct" Structure-Symbol "(#" Selector-List "#)" ";".
Selector-List = Selector-Segment { "," Selector-Segment }.
Selector-Segment =
     Selector-Symbol { "," Selector-Symbol } Specification .
Function-Loci = Locus |"(" Loci ")".
Loci = Locus { "," Locus }.
Locus = Variable-Identifier.

Specification = "->" Type .

Definiens = Simple-Definiens | Compound-Definiens  .

Simple-Definiens = Sentence .
Compound-Definiens = Partial-Definiens-List [ "otherwise" Sentence ] .
Partial-Definiens-List =
     Partial-Definiens { "," Partial-Definiens } .
Partial-Definiens = Sentence "if" Sentence .

Scheme =
     "scheme" Scheme-Identifier "{"  Scheme-Parameter-List "}" ":"
```

```
          Scheme-Conclusion
            "provided" Scheme-Premise { "and" Scheme-Premise }
          Justification ";" .
Scheme-Conclusion = Sentence.
Scheme-Premise = Proposition .
Scheme-Parameter-List = Scheme-Parameter { "," Scheme-Parameter } .
Scheme-Parameter =
     Local-Function-Pattern Specification |
     Local-Predicate-Pattern .

Local-Function-Pattern =
     Function-Identifier "(" [ Type-List ] ")" .
Local-Predicate-Pattern =
     Predicate-Identifier "[" [ Type-List ] "]" .

Generalization = "let" Fixed-Variables .
Assumption =
     Single-Assumption |
     Collective-Assumption |
     Existential-Assumption .
Single-Assumption = "assume" Sentence ";" .
Collective-Assumption = "assume" Conditions ";" .
Existential-Assumption = "given" Fixed-Variables ";" .

Compact-Statement = Sentence ";" .

Fixed-Variables = Qualified-Variables [ "such" Conditions ] .
Conditions = "that" Sentence { "and" Sentence } .
Proposition = [ Label-Identifier ":" ] Sentence .

Sentence = Formula .
Formula =
     Atomic-Formula |
     Quantified-Formula |
     Formula "&"  Formula |
     Formula "or"  Formula |
     Formula "implies"  Formula |
     Formula "iff"  Formula |
     "not"  Formula |
     "contradiction" .
Quantified-Formula =
     "for" Qualified-Variables [ "st" Formula ]
```

```
          ( "holds" Formula | Quantified-Formula ) |
     "ex" Qualified-Variables "st" Formula .

Atomic-Formula =
      [ Term-List ] Predicate-Symbol [ Term-List ] |
     Term ( "<>" | "=" ) Term |
     Predicate-Identifier "[" [ Term-List ] "]" |
     Term "is" Type .
Qualified-Variables =
     Implicitly-Qualified-Variables |
     Explicitly-Qualified-Variables |
     Explicitly-Qualified-Variables ","
          Implicitly-Qualified-Variables .
Explicitly-Qualified-Variables =
     Qualified-Segment { "," Qualified-Segment } .
Qualified-Segment = Variable-List Qualification .
Implicitly-Qualified-Variables = Variable-List .

Variable-List =
     Variable-Identifier {"," Variable-Identifier } .
Qualification =  ("being" | "be" ) Type .

Type =    "(" Type ")" |
            Mode-Symbol [ "of" Term-List ] |
            Structure-Symbol |
            "set" [ "of" Type ] |
            "[" Type-List "]" .
Type-List = Type { "," Type } .

Term = "(" Term ")" |
     [ Argument-List ] Function-Symbol [ Argument-List ]  |
     Left-Function-Bracket Term-List Right-Function-Bracket |
     Function-Identifier "(" [ Term-List ] ")" |
     "the" Selector-Symbol "of" Term |
     "the" Selector-Symbol  |
     Structure-Symbol "," Term-List "." |
     Variable-Identifier |
     "[" Term-List "]" |
     "{" Term-List "}" |
     "{" Term ":" Sentence "}" |
     Numeral |
     "it"  |
```

```
     Term "qua" Type .
Term-List = Term { "," Term } .
Argument-List = Term | "(" Term-List ")" .

Variable-Identifier = Identifier .
Function-Identifier = Identifier .
Predicate-Identifier = Identifier .
Scheme-Identifier = Identifier .
Label-Identifier = Identifier .

Vocabulary-File-Name = File-Name .
Signature-File-Name = File-Name .
Definitions-File-Name = File-Name .
Theorems-File-Name = File-Name .
Schemes-File-Name = File-Name .

File-Name = Identifier .

Structure-Symbol = Symbol .
Selector-Symbol = Symbol .
Predicate-Symbol = Symbol .
Function-Symbol = Symbol .
Mode-Symbol = Symbol .
Left-Function-Bracket = Symbol .
Right-Function-Bracket = Symbol .
```

Appendix C

Vocabularies

[ddd] stands for a character from extended ASCII with code ddd > 127 .

Vocabulary BIN_OP

BinOp	BinOp
UnOp	UnOp
the_unity_wrt	the unity wrt
is_associative	is associative
is_commutative	is commutative
is_a_unity_wrt	is a unity wrt
is_a_left_unity_wrt	is a left unity wrt
is_a_right_unity_wrt	is a right unity wrt
is_an_idempotent	is an idempotent
is_distributive_wrt	is distributive wrt
is_left_distributive_wrt	is left distributive wrt
is_right_distributive_wrt	is right distributive wrt

Vocabulary BOOLE

U	$\cup$
\	$\setminus$
c=	$\subseteq$
[237]	$\emptyset$
[239]	$\cap$
[246]	$\dot{-}$
meets	meets
misses	misses

Vocabulary BOOLEDOM

BOOLE_DOMAIN	BOOLE DOMAIN

Vocabulary COLLAPS

M[230]	Mμ
is_[238]-isomorphism_of	is $\in$-isomorphism of
are_[238]-isomorphic	are $\in$-isomorphic

Vocabulary COORD

‘1	^{1}
‘2	^{2}
‘3	^{3}
‘4	^{4}

Vocabulary EQUI_REL

[247]	$\approx$

Vocabulary FAM_OP

meet	$\bigcap$
union	$\bigcup$

Vocabulary FINITE

Fin	Fin
is_finite	is finite
Finite_Subset	Finite Subset

Vocabulary FINSEQ

FinSequence	FinSequence
FinSubsequence	FinSubsequence
Seg	Seg
len	len
^	$\frown$
Seq	Seq
Sgm	Sgm

`*`	$\star$
`<[237]>`	ε
`<*`	$\langle$
`*>`	$\rangle$

Vocabulary FUNC

`graph`	graph
`id`	Id
`.`	.
`Function`	Function
`is_one-to-one`	is 1-1

Vocabulary FUNC2

`Funcs`	Funcs
`Permutation`	Permutation

Vocabulary FUNC3

`pr1`	π_1
`pr2`	π_2
`delta`	δ
`incl`	incl
`chi`	χ
`<:`	$[\!($
`:>`	$)\!]$

Vocabulary FUNC_REL

`dom`	dom
`rng`	rng
`|`	$\restriction$
[248]	·

Vocabulary HIDDEN

`Any`	Any
`Element`	Element
`DOMAIN`	DOMAIN
`TUPLE`	TUPLE

`Subset`	Subset
`SUBDOMAIN`	SUBDOMAIN
`Real`	Real
`Nat`	Nat
`bool`	bool
`REAL`	REAL
`set`	set
`NAT`	NAT
`SET_DOMAIN`	SET DOMAIN
`[:`	⟦
`:]`	⟧
`+`	$+$
238	$\in$
243	$\leqslant$
249	$\cdot$

Vocabulary INCSP_1

`IncStruct`	IncStruct
`Points`	Points
`Lines`	Lines
`Planes`	Planes
`Inc1`	Inc1
`Inc2`	Inc2
`Inc3`	Inc3
`on`	on
`is_collinear`	is collinear
`is_coplanar`	is coplanar
`POINT`	POINT
`LINE`	LINE
`PLANE`	PLANE
`IncSpace`	IncSpace
`Line`	Line
`Plane`	Plane

Vocabulary LATTICES

`Lattice`	Lattice
`D_Lattice`	D Lattice
`M_Lattice`	M Lattice
`0_Lattice`	0 Lattice
`1_Lattice`	1 Lattice

01_Lattice	01 Lattice
C_Lattice	C Lattice
B_Lattice	B Lattice
[243][243]	$\sqsubseteq$
is_comp	is a complement
[192][217]	$\sqcup$
[218][191]	$\sqcap$
[193]	$\perp$
[194]	$\top$
LattStr	LattStr
L_carrier	L carrier
L_join	L join
L_meet	L meet

Vocabulary NAT_1

[179]	$\mid$
mod	mod
div	$\div$
lcm	lcm
hcf	gcd

Vocabulary ORDINAL

succ	succ
zero	$\mathbf{0}$
is_[238]-transitive	is $\in$-transitive
is_[238]-connected	is $\in$-connected
is_limit_ordinal	is limit ordinal
Ordinal	Ordinal
T-Sequence	transfinite sequence

Vocabulary REAL_1

-	$-$
"	$^{-1}$
/	$/$
<	$<$

Vocabulary REL_REL

is_reflexive_in	is reflexive in
is_irreflexive_in	is irreflexive in
is_symmetric_in	is symmetric in
is_antisymmetric_in	is antisymmetric in
is_asymmetric_in	is asymmetric in
is_connected_in	is connected in
is_strongly_connected_in	is strongly connected in
is_transitive_in	is transitive in
is_reflexive	is reflexive
is_irreflexive	is irreflexive
is_symmetric	is symmetric
is_antisymmetric	is antisymmetric
is_asymmetric	is asymmetric
is_connected	is connected
is_strongly_connected	is strongly connected
is_transitive	is transitive

Vocabulary RELATION

Relation	Relation
empty	$\emptyset$
field	field
diagonal	$\triangle$
~	$\smile$

Vocabulary SFAMILY

Set-Family	Set-Family
Subset-Family	Subset-Family
is_finer_than	is finer than
is_coarser_than	is coarser than
UNION	$\Cup$
INTERSECTION	$\Cap$
DIFFERENCE	$\setminus\setminus$

Vocabulary SUB_OP

234	Ω
`	c

Vocabulary TOP1

Int	Int
is_domain	is domain
is_closed_domain	is closed domain
is_open_domain	is open domain
is_dense	is dense
is_nowheredense	is nowheredense
is_boundary	is boundary

Vocabulary TOPCON

Cl	Cl
Fr	Fr
skl	skl
carrier	carrier
topology	topology
TopStruct	TopStruct
is_open	is open
is_closed	is closed
is_open_closed	is open closed
are_separated	are separated
is_continuous	is continuous
are_joined	are joined
is_a_component_of	is a component of
is_a_cover_of	is a cover of
TopSpace	TopSpace
Point	Point
SubSpace	SubSpace
map	map

Vocabulary WELLORD

is_well_founded_in	is well founded in
is_well_founded	is well founded
well_orders	well orders
is_well-ordering-relation	is well-ordering-relation
are_isomorphic	are isomorphic
is_isomorphism_of	is isomorphism of
-Seg	-Seg
$\mid$ 253	$\restriction^2$
canonical_isomorphism_of	canonical isomorphism of

Vocabulary ZF_AXIOM

the_axiom_of_extensionality	the axiom of extensionality
the_axiom_of_pairs	the axiom of pairs
the_axiom_of_unions	the axiom of unions
the_axiom_of_infinity	the axiom of infinity
the_axiom_of_power_sets	the axiom of power sets
the_axiom_of_substitution_for	the axiom of substitution for

Vocabulary ZF_LANG

Variable	Variable
ZF-formula	ZF-formula
'='	'$=$'
'238'	'$\in$'
170	$\neg$
'&'	$\wedge$
All	$\forall$
'or'	$\vee$
205>	$\Rightarrow$
<205>	$\Leftrightarrow$
Ex	$\exists$
WFF	WFF
VAR	VAR
x.	ξ
Subformulae	Subformulae
Var1	Var_1
Var2	Var_2
the_argument_of	the argument of
the_left_argument_of	the left argument of
the_right_argument_of	the right argument of
the_scope_of	the scope of
bound_in	bound in
the_antecedent_of	the antecedent of
the_consequent_of	the consequent of
the_left_side_of	the left side of
the_right_side_of	the right side of
is_immediate_constituent_of	is immediate constituent of
is_subformula_of	is subformula of
is_proper_subformula_of	is proper subformula of
is_equality	is equality
is_membership	is membership
is_atomic	is atomic
is_negative	is negative

is_conjunctive	is conjunctive
is_universal	is universal
is_disjunctive	is disjunctive
is_conditional	is conditional
is_biconditional	is biconditional
is_existential	is existential

Vocabulary ZF_SAT

Free	Free
VAL	VAL
St	St
199 196	$\models$
is_a_model_of_ZF	is a model of ZF

Bibliography

[1] R. Constable et al. *Implementing Mathematics with the Nuprl Proof Development Syst em.* Prentice-Hall, 1986.

[2] N. G. de Bruijn. A survey of the project AUTOMATH. In J. P. Seldin and J. R. Hindley, editors, Essays in Combinatory Logic, Lambda Calculus, and Formalism, pages 589–606. Academic Press, 1980.

[3] J. Ketonen. EKL—a mathematically oriented proof checker. In *Proceedings of 7th Int. Conf. on Automated Deduction*, pages 65–79, Napa, CA, May 1984, 1973.

[4] D. E. Knuth. *The TEXbook.* Addison Wesley, 1984.

[5] L. Lamport. *LATEX: a document preparation system.* Addison Wesley, 1985.

[6] M. Mostowski and Z. Trybulec. A certain experimental computer aided course of logic in Poland. In *Proceedings of World Conference on Computers in Education.* Norfolk, VA, 1985.

[7] Y. Nakamura. A language for description of mathematics—THEAX. Technical report, Shinshu University FIE, Nagano City, Japan, 1985. In Japanese.

[8] D. Nieva Soto. The reasoner of MIZAR/LOG. *Computerized Logic Teaching Bulletin*, 2(1):22–35, 1989.

[9] K. Prażmowski, P. Rudnicki, et al. Mizar-MSE primer and user guide. TR 88–9, University of Alberta, Department of Computing Science, 1988.

[10] P. Rudnicki. Obvious inferences. *Journal of Automated Reasoning*, 3:383–393, 1987.

[11] P. Rudnicki and W. Drabent. Proving properties of Pascal programs in Mizar-2. *Acta Informatica*, 22(3):311–331 and 699–707, 1985.

[12] R. Smith et al. Computer-assisted axiomatic mathematics: Informal rigor. In O. Lecarme and R. Lewis, editors, *Computers in Education*, pages 803–809. North Holland, 1975.

[13] A. Trybulec. Mizar syntax. *Studies in Logic, Grammar, and Rhetorics*, 6, 1994. To appear.

[14] A. Trybulec and H. Blair. Computer aided reasoning. In R. Parikh, editor, *Logic of Programs, LNCS 193*. Springer Verlag, 1985.

[15] A. Trybulec and H. Blair. Computer assisted reasoning with Mizar. In *Proceedings of the 9th IJCAI*, pages 26–28, 1985.