A Language for Mathematical Knowledge Management

Steve Kieffer

September 2, 2007

Acknowledgements

This work would not have been possible without the patience and guidance of my advisor Jeremy Avigad, or without ideas which came both from him and from Harvey Friedman.

I am also grateful for thoughtful comments and revisions from Joe Ramsey.

Thanks also to my parents, who were indispensable in preparing and practicing a talk on this material.

It is dedicated to my brother Matt, who would have insisted this be published in fanzine format if he were still with us today.

Contents

Chapter 1

Introduction

We live at a time at which, on the one hand, mathematical research is producing more and more knowledge; and on the other hand the abilities of computers to manage data are becoming better and better understood. It is natural then that in the last couple of decades efforts have been launched in a field called *Mathematical Knowledge Management* (MKM), in which computer systems are being designed for a number of purposes pertaining to mathematical knowledge. The field is represented each year at the conference on MKM which has met annually since 2002.

While the 'management' in the name of the field is perhaps most directly suggestive of database maintenance, the field is properly understood to include projects with a wider variety of purposes than this. Others include the presentation and visualization of mathematical knowledge, interactive instruction in mathematics, and the formal verification of mathematical knowledge. This breadth of subjects is reflected in the variety of papers collected each year in the MKM conference proceedings [4].

Meanwhile efforts are underway in the philosophy of mathematics to clarify the nature of, and differences between, such things as mathematical *understanding*, *skill*, and *knowledge*. (See [6] for example, and references cited therein.) While philosophers working in this area continue to clarify what it is that mathematical knowledge consists of, we may certainly for the time being take *theorems*, and *defined concepts* to constitute at least part of it. We may therefore direct our present efforts in MKM toward the design of systems to manage these two sorts of mathematical product. In this project, I focus on definitions.

In order to use computer systems to store, retrieve, manipulate, measure for complexity, or even depict, the content of mathematical definitions and theorems, and in order to computationally verify theorems, we need a suitable *language* in which the definitions and theorems can be written. The central contribution of the present

thesis project is the presentation of such a language, along with the development of a computer system to parse input written in this language, and store it in a form amenable to further use.

As peripheral contributions I have designed three other computer systems serving MKM purposes, each of which utilizes the data stored by the first system. I discuss them in greater detail below. Briefly, they are (1) A translator, translating the input language to the language of a system of set theory; (2) Programs that gather various structural data from definitions; and (3) A graphical user interface allowing the user to explore the conceptual dependencies among defined concepts.

By way of testing the parser, I have compiled a database of 341 definitions, which can be found in Appendices F and G. These are formalized versions of the definitions appearing in two textbooks: (1) Chapters 2 through 6 of Suppes's *Axiomatic Set Theory* [15]; and (2) Sections 12 through 38 of Munkres's *Topology* [12]. I did not formalize any theorems, as the definitions were enough to meet the goals of this project.

Likewise, the database was assembled just for the purposes of the immediate project, and is not expected to be usable by others, at least not in its present form. This made it possible to bracket important database and search issues, for the time being.

In the future, however, we may have to consider such issues, for example, searchability. Keyword search will certainly be possible, if we keep the database in its present form. On the other hand, semantic search would be desirable too, and for this we would benefit by augmenting the current definition syntax to allow for semantic tags, indicating subject matter.

A serious attempt to develop a definition library would also prompt better naming conventions. For one thing we might want to allow the database to be distributed across servers, and in that case namespaces would be a good idea. Even if not, it would be wise to set down rules for naming definitions in a way that reflects their sources (as I have done in the definitions appearing in Appendices F and G).

Finally, we have also deferred questions as to whether the definitions in our database could be interchangeable with other MKM systems.

The input language considered in this project was designed by Harvey Friedman. With the view in mind that written mathematics consists of theorems, definitions, and proofs, and with the goal in mind of designing a language for the recording of just theorems and definitions, Friedman designed a formal system called "Proofless Text." The language we consider is essentially the language of this system (with minor variations); accordingly I refer to it as *the language of proofless text*, or LPT for short. (The reason for leaving out proofs at this stage is that a language for proofs

requires more structure than is needed for definitions and theorems. Such languages can be designed – and many good ones have been – but this was not necessary for our purposes in this paper.)

There are many things to consider when choosing a language for MKM. In order to allow for the automated manipulation, storage, and retrieval of information, we should use a language that can be processed by computer programs. On the other hand, we want a language that is easily readable by human beings.

LPT is designed to be highly readable, and yet completely formal. The readability comes from the inclusion in LPT of special features to accommodate certain types of locutions that are used over and over again by mathematicians when they state theorems, or define concepts. A recent language presented at the MKM conference [7] includes set-builder notation (and other notations introduced as abbreviations). LPT includes this, as well as the following features, which are discussed in Section 2.2: large character set, infix functions and relations, arbitrary terms serving as functions and relations, tuple notation, description operator, lambda abstraction, qualified quantification, and term binding (as opposed to mere variable binding).

A few of these features can be seen working in combination in the following examples. First consider the following formalization in LPT of the definition of *the topology generated by a basis*, from Chapter 13 of Munkres's *Topology* [12]. We have it first in raw input form:

```
DEFINITION MunkTop.13.2: 2-ary function Basisgentop.
If TOPBASIS[\mathscr{B},X] then
Basisgentop(\mathscr{B},X) \simeq (!\mathscr{T} \subseteq \wp(X))(
  (\forall U \subseteq X)(U \in \mathscr{T} \iff
    (\forall x \in U)(\exists B \in \mathscr{B})(
      x \in B \wedge B \subseteq U
    )
  )
).
```

and also in its LaTeX version:

DEFINITION MunkTop.13.2: 2-ary function `Basisgentop`. If `TOPBASIS`$[\mathscr{B}, X]$ then `Basisgentop`$(\mathscr{B}, X) \simeq (!\mathscr{T} \subseteq \wp(X))((\forall U \subseteq X)(U \in \mathscr{T} \leftrightarrow (\forall x \in U)(\exists B \in \mathscr{B})(x \in B \wedge B \subseteq U)))$.

Here we see the description operator '!' applied to the variable $\mathscr{T}$, while this variable is simultaneously qualified by the clause '$\subseteq \wp(X)$' to be a subset of the power set of X.

In a second example from the same chapter of Munkres, we have the definition of a basis for the standard topology on the reals:

```
DEFINITION MunkTop.13.3.a.basis: 0-ary function Stdrealtopbasis.
Stdrealtopbasis \simeq {U \subseteq \mathbb{R} :
  (\exists a,b \in \mathbb{R})(
    U = {x \in \mathbb{R} : a <_{R} x <_{R} b}
  )
}.
```

DEFINITION MunkTop.13.3.a.basis: 0-ary function `Stdrealtopbasis`.
`Stdrealtopbasis` $\simeq \{U \subseteq \mathbb{R} : (\exists a, b \in \mathbb{R})(U = \{x \in \mathbb{R} : a <_{\mathtt{R}} x <_{\mathtt{R}} b\})\}$.

Here we see use of a defined infix relation $<_R$. We also see use of set-builder notation, in which we can name "the set of all $x \in \mathbb{R}$ ...," not merely "the set of all x ...." These and the other special features of LPT are covered in Chapter 2.

Naturally what would be most pleasing to the eye of English speaking users of computer MKM systems would be to work in the English language itself, with mathematical symbols thrown in. But this might not be the best method after all. Users want to have some certainty that their input has been processed correctly by the computer. If they should write a definition using an English sentence that has an ambiguous syntax tree, how can they be confident that the ambiguity has been correctly resolved (as by reference to semantic knowledge in an artificial intelligence system)?

Perhaps there would have to be some reformulation, produced by the computer, that the user could look at and check that everything was as it should be. What form would this take? Suppose the natural language input were to be recast by the computer into a more formal language, and sent back to the user for approval. The user then would have to be able to read this language. But if the user became fluent in the formal language, then he or she might just as well write the input in it directly.

Perhaps then some highly readable, and yet formal, language is best. LPT is an offering in this regard, and users may try it out and voice their approval or disapproval. I can already envision improvements: Much of the niceness of natural language could probably be incorporated without involving any syntactic ambiguity. Even something as simple as replacing formal predication (i.e. predication in functional form, as in "$P[x, y, z]$") by its natural language equivalent would be a major improvement.

In the end, there are better and worse, more and less user-friendly input/output languages to be designed. I am presenting one. The Mizar system offers another [1].

There are many more to be found in the MKM literature. We seem to be working at a time at which it makes sense to try out many different languages and computer systems for MKM, let users try them out, and see what works best, see what people like best.

At this time it is still tricky to design systems that use natural language, or fragments thereof. By opting for a simpler system, I have been able to get a system up and running, and have been able to do some satisfying things with it.

To begin with, there is the database of definitions mentioned above. These definitions are written in LPT. Those taken from Suppes [15] bring us from set-theoretic foundations up through the natural, rational, and real numbers. Those taken from Munkres [12] bring us from basic point-set topology up through connectedness and compactness, the countability and separation axioms, and the Stone-Čech compactification.

In Chapter 2 I describe the language LPT, go over its special features, and discuss the computer system I have designed to parse LPT input.

As a first application of my database of definitions, I have written a program to translate from LPT into the language of DZFC (or "Definitional ZFC", a system that adds definitional capabilities to ordinary Zermelo-Frankael set theory with Axiom of Choice). In Chapter 3 I discuss the translation system. I also prove that DZFC is a conservative extension of ZFC, and is therefore a suitable system into which to translate the LPT input.

As a second application of my database of definitions, I have gathered structural data, and thus have begun to meet one of a list of goals enunciated by Harvey Friedman in [10]. Friedman poses the rhetorical question as to why we should want to design a system in which formalization is *convenient* (we already know it is *possible*). In answer he names several purposes, one of which is:

> To obtain detailed information about the logical structure of mathematical concepts. For instance, what are the appropriate measures of the depth or complexity of mathematical concepts?

In answer, I have investigated several measures of the logical structure of the definitions in my database. To begin with, I have determined for each definition the depth of, and number of nodes in, its "definition DAG" (Directed Acyclic Graph). In this I have followed Friedman and Flagg, who define the definition DAG in [9]; roughly, it is a graph of dependencies of one defined concept on another. I give the precise formulation in Chapter 4.

Also in Chapter 4, I examine the quantifier nesting depth for each definition. I consider this first with the understanding that depth increases only when there

is a quantifier alternation, and then again where there need not be an alternation. I also regard each definition in four different ways: (1) as given in LPT; (2) as translated into DZFC; (3) the result of fully "expanding" the DZFC version, i.e., the result of replacing every definiendum by its definiens, recursively, until no definienda remain; and finally (4) similarly expanded, only with a certain set of definienda left unexpanded. For the latter I chose a list of certain low, foundational definitions (e.g. the set theoretic definition of the ordered pair), whose expansion seemed to make an inordinate contribution to the complexity of the expanded definition.

I finish out Chapter 4 with two more structural observations. For one, I consider the raw symbol count of definitions, again regarded in various states of expansion, as above. For another, I indicate what sorts of functions and relations, i.e. what arities, were defined in the database, and I tell how many of each kind there were.

Finally, I have developed a computer program with a graphical user interface (GUI), that allows the user to selectively expand and collapse the definition DAG for any definition in my database. I discuss the program in Chapter 5. I foresee the future refinement of this program, and hope to see the program used by students of mathematics when exploring fields new to them.

Chapter 2

LPT, the Language of Proofless Text

The readability of LPT comes from the special characters and strings it provides for the naming of variables, functions, and relations; and also from the special types of terms and formulas that can be formed in it. In the following section I give a precise, inductive definition of the terms and formulas of LPT, as well as a specification of the syntax for writing definitions. In the next section I discuss the special features of the language at greater length, and give examples. Finally, in the last section I cover the method used to parse the language.

2.1 Syntax

2.1.1 Terms and Formulas

Definitions in LPT are definitions of *functions*, and *relations*. Accordingly, the language features an infinite collection of strings to serve as the names of these functions and relations. The formation of these strings, and the character sets on which they draw, are detailed in Appendix A. This presentation follows Friedman [11].

Each function and each relation is defined with an arity, and therefore we speak of *applied function strings* and *applied relation strings*. An applied function string is a pair (x, k) or (x, infix), where x is a function string and $k \geq 0$. An applied relation string is a pair (x, k) or (x, infix), where x is a relation string and $k \geq 1$.

Since a defined function or relation can be used in later definitions, the legal expressions in LPT at any given time depend on the collection of functions and relations

that have so far been defined, and their arities. If σ is the set of applied function strings and applied relation strings so far defined, then the terms and formulas that are well-formed at that time are called the σ terms and σ formulas.

Since we use infix functions and relations, and infix logical connectives, it is sometimes unambiguous where a term or formula begins and ends, and sometimes not; in the former case we call the σ term or σ formula *bracketed*.

We simultaneously define the σ terms, bracketed σ terms, σ formulas, and bracketed σ formulas in the twelve enumerated points below.

1. Every variable string (see Appendix A) is a bracketed σ term. Every decimal numeral is a bracketed σ term. `\bot` and `\top` are bracketed σ formulas.

2. Let $k \geq 1$ and $s, t_1, \ldots, t_k$ be σ terms. Let r be a bracketed σ term. Then

   $$(s)$$
   $$\{t_1, \ldots, t_k\}$$
   $$\langle t_1, \ldots, t_k \rangle$$
   $$r(t_1, \ldots, t_k)$$

   are bracketed σ terms.

3. Let $k \geq 1$ and $s, t_1, \ldots, t_k$ be σ terms. Then

   $$s\uparrow$$
   $$s\downarrow$$
   $$s[t_1, \ldots, t_k]$$

   are bracketed σ formulas.

4. Let α be an applied function string in σ with arity 0. Then α is a bracketed σ term.

5. Let α be an applied function string in σ with arity $k \geq 1$. Let $s_1, \ldots, s_k$ be σ terms. Then $\alpha(s_1, \ldots, s_k)$ is a bracketed σ term.

6. Let $k \geq 2$. For each $1 \leq i \leq k-1$, let α_i be either

   (i) an infix function string in σ; or

   (ii) an expression of the form `\infixfn`$\{r\}$, where r is a σ term.

Let $s_1, \ldots, s_k$ be bracketed σ terms. Then

$$s_1\ \alpha_1\ s_2\ \cdots\ \alpha_{k-1}\ s_k$$

is a σ term.

7. Let β be an applied relation string in σ with arity $k \geq 1$. Let $s_1, \ldots, s_k$ be σ terms. Then $\beta[s_1, \ldots, s_k]$ is a bracketed σ formula.

8. Let $p \geq 2$ and $n_1, \ldots, n_p \geq 1$. For $1 \leq i \leq p$ and $1 \leq j \leq n_i$, let s_{ij} be a σ term. For all $1 \leq i < p$, let β_i be

   (i) an infix relation string in σ; or

   (ii) $=$, \neq, \simeq; or

   (iii) an expression of the form \infixrl{r}, where r is a σ term.

   Then

   $$s_{11}, \ldots, s_{1,n1}\ \beta_1\ s_{21}, \ldots, s_{2,n2}\ \cdots\ \beta_{p-1}\ s_{p1}, \ldots, s_{p,np}$$

   is a σ formula.

9. Let $k, n \geq 1$. Let $v_1, \ldots, v_k$ be distinct variables, $t, s_1, \ldots, s_n$ be σ terms, φ be a bracketed σ formula, and ψ be a σ formula. Let α be

   (i) an infix relation string in σ; or

   (ii) $=$, \neq, \simeq; or

   (iii) an expression of the form \infixrl{r}, where r is a σ term.

   Then

   $$(!t)\varphi$$
   $$(!t)(\psi,\ v_1, \ldots, v_k \text{ fixed})$$
   $$(!t\ \alpha\ s_1, \ldots, s_n)\varphi$$
   $$(!t\ \alpha\ s_1, \ldots, s_n)(\psi,\ v_1, \ldots, v_k \text{ fixed})$$

   $$\{t :\ \psi\}$$
   $$\{t :\ \psi,\ v_1, \ldots, v_k \text{ fixed}\}$$
   $$\{t\ \alpha\ s_1, \ldots, s_n :\ \psi\}$$
   $$\{t\ \alpha\ s_1, \ldots, s_n :\ \psi,\ v_1, \ldots, v_k \text{ fixed}\}$$

   are bracketed σ terms.

10. Let $k, n \geq 1$. Let $v_1, \ldots, v_k$ be distinct variables, $s_1, \ldots, s_n$ be σ terms, t be a bracketed σ term, and ψ be a σ formula. Let α be

    (i) an infix relation string in σ; or

    (ii) $=$, \neq, \simeq; or

    (iii) an expression of the form \infixrl{r}, where r is a σ term.

    Then

    $$(\lambda v_1, \ldots, v_k)t$$
    $$(\lambda v_1, \ldots, v_k : \psi)t$$
    $$(\lambda v_1, \ldots, v_k\ \alpha\ s_1, \ldots, s_n)t$$
    $$(\lambda v_1, \ldots, v_k\ \alpha\ s_1, \ldots, s_n : \psi)t$$

    are bracketed σ terms.

11. Let $k \geq 2$ and $\varphi, \rho_1, \ldots, \rho_k$ be σ formulas. Let ψ be a bracketed σ formula. Let $\alpha_1, \ldots, \alpha_{k-1}$ be among $\vee$, $\wedge$, $\rightarrow$, $\leftrightarrow$. Then

    (i) (φ)

    (ii) $\neg\psi$

    (iii) $\rho_1\ \alpha_1\ \rho_2\ \cdots\ \alpha_{k-1}\ \rho_k$

    (iv) $![\rho_1, \ldots, \rho_k]$

    are σ formulas. The first two and the fourth are bracketed σ formulas.

12. Let $k, n, m \geq 1$. Let $v_1, \ldots, v_k$ be distinct variables, and $t_1, \ldots, t_n, s_1, \ldots, s_m$ be σ terms. Assume that $t_1, \ldots, t_n$ are distinct. Let α be

    (i) an infix relation string in σ; or

    (ii) $=$, \neq, \simeq; or

    (iii) an expression of the form \infixrl{r}, where r is a σ term.

Let φ be a bracketed σ formula and ψ, ρ be σ formulas. Then

$(\forall t_1, \ldots, t_n)\varphi$
$(\exists t_1, \ldots, t_n)\varphi$
$(\exists! t_1, \ldots, t_n)\varphi$
$(\forall t_1, \ldots, t_n\ \alpha\ s_1, \ldots, s_m)\varphi$
$(\exists t_1, \ldots, t_n\ \alpha\ s_1, \ldots, s_m)\varphi$
$(\exists! t_1, \ldots, t_n\ \alpha\ s_1, \ldots, s_m)\varphi$
$(\forall t_1, \ldots, t_n)(\psi,\ v_1, \ldots, v_k \text{ fixed})$
$(\exists t_1, \ldots, t_n)(\psi,\ v_1, \ldots, v_k \text{ fixed})$
$(\exists! t_1, \ldots, t_n)(\psi,\ v_1, \ldots, v_k \text{ fixed})$
$(\forall t_1, \ldots, t_n\ \alpha\ s_1, \ldots, s_m)(\psi,\ v_1, \ldots, v_k \text{ fixed})$
$(\exists t_1, \ldots, t_n\ \alpha\ s_1, \ldots, s_m)(\psi,\ v_1, \ldots, v_k \text{ fixed})$
$(\exists! t_1, \ldots, t_n\ \alpha\ s_1, \ldots, s_m)(\psi,\ v_1, \ldots, v_k \text{ fixed})$
$(\forall t_1, \ldots, t_n : \rho)\varphi$
$(\exists t_1, \ldots, t_n : \rho)\varphi$
$(\exists! t_1, \ldots, t_n : \rho)\varphi$

are bracketed σ formulas.

2.1.2 Definitions

There are four basic types of definition in LPT, corresponding to two binary choices: First, a definition is either of a function or of a relation; second, the arity of the defined function or relation is either "infix", or an ordinary numerical arity (a non-negative integer). Furthermore, definitions will take on different forms according to the number of clauses that they use. In LPT a piecewise definition can be given, using `If ... then ...` clauses, as well as an optional `Otherwise ...` clause.

If the definiendum α is a *relation* string, then the definition takes on one of the six following forms.

1. `DEFINITION` β: `k-ary relation` α. $\alpha[v_1, \ldots, v_k] \leftrightarrow \psi$.
2. `DEFINITION` β: `k-ary relation` α. `If` φ_1 `then` $\alpha[v_1, \ldots, v_k] \leftrightarrow \psi_1$. ... `If` φ_n `then` $\alpha[v_1, \ldots, v_k] \leftrightarrow \psi_n$.
3. `DEFINITION` β: `k-ary relation` α. `If` φ_1 `then` $\alpha[v_1, \ldots, v_k] \leftrightarrow \psi_1$. ... `If` φ_n `then` $\alpha[v_1, \ldots, v_k] \leftrightarrow \psi_n$. `Otherwise` $\alpha[v_1, \ldots, v_k] \leftrightarrow \psi_{n+1}$.
4. `DEFINITION` β: `Infix relation` α. $v\ \alpha\ w \leftrightarrow \psi$.

5. DEFINITION β: Infix relation α. If φ_1 then $v\ \alpha\ w \leftrightarrow \psi_1$. ... If φ_n then $v\ \alpha\ w \leftrightarrow \psi_n$.

6. DEFINITION β: Infix relation α. If φ_1 then $v\ \alpha\ w \leftrightarrow \psi_1$. ... If φ_n then $v\ \alpha\ w \leftrightarrow \psi_n$. Otherwise $v\ \alpha\ w \leftrightarrow \psi_{n+1}$.

For the first three forms, $k, n \geq 1$, $v_1, \ldots, v_k$ are distinct variables, and $\varphi_1, \ldots, \varphi_n, \psi, \psi_1, \ldots, \psi_{n+1}$ are bracketed σ formulas whose free variables are among $v_1, \ldots, v_k$.

In the last three forms, $n \geq 1$, v, w are distinct variables, and $\varphi_1, \ldots, \varphi_n, \psi, \psi_1, \ldots, \psi_{n+1}$ are bracketed σ formulas whose free variables are among v, w.

If the definiendum α is a function string, then the definition takes on one of the seven following forms.

1. DEFINITION β: 0-ary function α. $\alpha \simeq t$.

2. DEFINITION β: k-ary function α. $\alpha(v_1, \ldots, v_k) \simeq t$.

3. DEFINITION β: k-ary function α. If φ_1 then $\alpha(v_1, \ldots, v_k) \simeq t_1$. ... If φ_n then $\alpha(v_1, \ldots, v_k) \simeq t_n$.

4. DEFINITION β: k-ary function α. If φ_1 then $\alpha(v_1, \ldots, v_k) \simeq t_1$. ... If φ_n then $\alpha(v_1, \ldots, v_k) \simeq t_n$. Otherwise $\alpha(v_1, \ldots, v_k) \simeq t_{n+1}$.

5. DEFINITION β: k-ary function α. $v\ \alpha\ w \simeq t$. Precedence r.

6. DEFINITION β: k-ary function α. If φ_1 then $v\ \alpha\ w \simeq t_1$. ... If φ_n then $v\ \alpha\ w \simeq t_n$. Precedence r.

7. DEFINITION β: k-ary function α. If φ_1 then $v\ \alpha\ w \simeq t_1$. ... If φ_n then $v\ \alpha\ w \simeq t_n$. Otherwise $v\ \alpha\ w \simeq t_{n+1}$. Precedence r.

In any of the forms 2 through 7, any clause of the form $\alpha(v_1, \ldots, v_k) \simeq t$ or $v\ \alpha\ w \simeq t$ may be replaced by $\alpha(v_1, \ldots, v_k) \uparrow$ or $v\ \alpha\ w \uparrow$, respectively, where the upward arrow $\uparrow$ indicates that the function is undefined.

In the first four forms, $k, n \geq 1$, and $v_1, \ldots, v_k$ are distinct variables; $\varphi_1, \ldots, \varphi_n$ are bracketed σ formulas, and t, $t_1, \ldots, t_{n+1}$ are σ terms in which all free variables are among $v_1, \ldots, v_k$.

In the final three forms, $n \geq 1$, and v, w are distinct variables; $\varphi_1, \ldots, \varphi_n$ are bracketed σ formulas, and t, $t_1, \ldots, t_{n+1}$ are σ terms in which all free variables are among v, w.

The precedence number r is either a finite string of digits, or a finite string of digits with a minus sign in front. Each infix function is assigned a precedence number so that the translator program can assign the appropriate order of operations in the case of a sequence of ambiguous infix function applications. When comparing precedence numbers, any negative number comes before zero, and zero comes before any positive number. Among negative or positive numbers, the lexicographic ordering is used.

2.2 Special Features

The special features of LPT can be seen in the formal presentation given in the last section. Here I elaborate on them, and give examples.

1. Infix functions and relations. Defined function and relation srings may be given a special "infix" arity. This means that they are binary, but when written are placed in between their two arguments.

In the first example below we have the definition of an equivalence relation that we use in defining fractions of natural numbers. It is defined as an infix relation.

```
DEFINITION FS.5.3: Infix relation \equiv_{Fr}. x \equiv_{Fr} y \iff
(\exists a,b,c,d)(
  x = a/b \wedge y = c/d \wedge a \cdot_{N} d = b \cdot_{N} c
).
```

DEFINITION FS.5.3: Infix relation $\equiv_{\mathtt{Fr}}$. $x \equiv_{\mathtt{Fr}} y \leftrightarrow (\exists a, b, c, d)(x = a/b \wedge y = c/d \wedge a \cdot_N d = b \cdot_N c)$.

In the next example we see the definition of addition on fractions, made as an infix function.

```
DEFINITION FS.5.8: Infix function +_{Fr}. x +_{Fr} y \simeq
(!z)(\exists a,b,c,d,e,f)(
  x = a/b \wedge y = c/d \wedge z = e/f \wedge
  e = a \cdot_{N} d +_{N} b \cdot_{N} c
  \wedge f = b \cdot_{N} d
). Precedence 40.
```

DEFINITION FS.5.8: Infix function $+_{\texttt{Fr}}$. $x +_{\texttt{Fr}} y \simeq (!z)(\exists a, b, c, d, e, f)(x = a/b \wedge y = c/d \wedge z = e/f \wedge e = a \cdot_N d +_N b \cdot_N c \wedge f = b \cdot_N d)$. Precedence 40.

2. Terms as functions and relations. Any term may be used as though it were a function or a relation, of any arity (including "infix"). For example, one may quantify a variable f, and then proceed to use it as though it were a function.

Below we have a definition in which the intention is to define the unary predicate `FCN` to assert of a set `f` that it is a set of ordered pairs `<x,y>` in which no `x` occurs more than once as the first component of a pair; that is, to assert that f is a function.

In LPT we can achieve this by first introducing f as a variable, and then using `f(x)` to mean the unique `u` such that the ordered pair `<x,u>` is in the set denoted by `f`, as below:

```
DEFINITION FS.2.58: 1-ary relation FCN. FCN[f] \iff
f = {<x,y> : f(x) = y}.
```

DEFINITION FS.2.58: 1-ary relation `FCN`. `FCN`$[f] \leftrightarrow f = \{\langle x, y\rangle : f(x) = y\}$.

In the next example we have a variable R being used as an infix relation.

```
DEFINITION FS.2.3: 1-ary function Dom. If BR[R] then Dom(R) \simeq
{x : (\exists y)(x \infixrl{R} y)}. Otherwise Dom(R) \up.
```

DEFINITION FS.2.3: 1-ary function `Dom`. If `BR`$[R]$ then `Dom`$(R) \simeq \{x : (\exists y)(xRy)\}$. Otherwise `Dom`$(R)\uparrow$.

It is intended that LPT be translated into systems that use "free logic"; that is, into systems in which there may be terms that fail to denote. (See [8] for example.) In the intended interpretation of LPT any term may fail to be meaningful as a function or relation of a particular arity. If a term fails to serve as a function, then the function application in question should simply fail to denote. If a term fails to serve as a relation, then the predication in question should simply be taken to be false.

3. Set builder notation. In LPT one can name a finite set by listing its elements explicitly. Thus, if $t_1, \ldots, t_n$ are terms then so is $\{t_1, \ldots, t_n\}$.

One can also name the set of all terms t that satisfy a formula ψ; thus, $\{t : \psi\}$ is a term in LPT. Note that t may be any term, and need not be a mere variable. Thus, for example, if f is a function one can name $\{f(x) : \psi\}$.

If α is an infix relation, t and $s_1, \ldots, s_n$ are terms, and ψ is a formula, then $\{t\,\alpha\,s_1, \ldots, s_n : \psi\}$ is a term. In the intended interpretation this is the set of all t standing in the relation α to each of the s_i, and also satisfying ψ. For example, one can name $\{x \in X : \psi\}$.

Here is an example of set builder notation in use, in defining the cartesian product of two sets:

```
DEFINITION FS.1.2: Infix function \times. x \times y \simeq {
  <z,w> : z \in x \wedge w \in y
}. Precedence 20.
```

DEFINITION FS.1.2: Infix function $\times$. $x \times y \simeq \{\langle z, w\rangle : z \in x \wedge w \in y\}$. Precedence 20.

4. Tuple notation. An ordered tuple of any length may be written using angle brackets. Thus, if $t_1, \ldots, t_n$ are terms then so is $\langle t_1, \ldots, t_n\rangle$. We see an example of tuple notation in the definition of the cartesian product, above.

5. Lambda notation. There is a lambda operator which can be used to bind variables and thereby denote functions, as in a lambda calculus. In the simplest case this takes the form $(\lambda v_1, \ldots, v_n)t$, where t is a term and the v_i are distinct variables.

One may also qualify the variables with a "such that" clause, and/or with an infix relation, just as may be done in denoting sets. Thus, if α is an infix relation, $s_1, \ldots, s_k$ are terms, and φ is a formula, then $(\lambda v_1, \ldots, v_n \alpha s_1, \ldots, s_k : \varphi)t$ is also a term.

In the example below, we define a binary function called `Cartespow` (for "Cartesian power"). This function maps a pair of sets A, B to the set A^B; i.e., a product of B-many copies of A.

This definition relies on a previously defined function, `Cartesprod` (for "Cartesian product"), a binary function taking a map f and a set C to the product over $c \in C$ of the sets $f(c)$.

The definition below used lambda abstraction to define a function on the fly, to serve as the first argument to `Cartesprod`.

```
DEFINITION MunkTop.19.2.5: 2-ary function Cartespow. Cartespow(A,B)
\simeq Cartesprod((\lambda b \in B)(A),B).
```

DEFINITION MunkTop.19.2.5: 2-ary function `Cartespow`. `Cartespow`$(A, B) \simeq$ `Cartesprod`$((\lambda b \in B)(A), B)$.

6. Unique existence, and qualified quantification. In addition to the usual $\forall$ and $\exists$ quantifiers we also include $\exists!$ for "there exists a unique".

One may quantify not just variables but terms in general. In the intended interpretation, quantification over terms simply means quantification over all the variables appearing in those terms.

Quantified terms may be further qualified, again, as in items above, with a "such that" clause, and/or with an infix relation.

For example, below we see a definition of what it means for a space to be *sequentially compact*. In it, we put a condition not merely on "all f," but rather on "all f in the set `Maps`(ω, X)."

```
DEFINITION MunkTop.28.3: 2-ary relation SEQCPT. If TOPSP[X,T] then
SEQCPT[X,T] \iff (\forall f \in Maps(\omega,X))
(\exists g : SUBSEQ[g,f])(
  (\exists x \in X)(TOPCONV[g,x,T])
).
```

DEFINITION MunkTop.28.3: 2-ary relation `SEQCPT`. If `TOPSP`$[X, T]$ then `SEQCPT`$[X, T] \leftrightarrow$ $(\forall f \in$ `Maps`$(\omega, X))(\exists g :$ `SUBSEQ`$[g, f])((\exists x \in X)($`TOPCONV`$[g, x, T]))$.

7. Description operator. Traditionally a description operator is another variable binder. We generalize this much as we have generalized the quantifiers: In LPT the description operator may be used to bind any term, not just a variable. Furthermore the term may be qualified with an infix relation.

Below we see a definition in which we have used the description operator ! to bind the ordered pair $\langle Y, T' \rangle$.

```
DEFINITION MunkTop.29.4: 2-ary function Oneptcompactification.
If TOPSP[X,T] then Oneptcompactification(X,T) \simeq
(!<Y,T'>)(
  COMPACTIFICATION[Y,T',X,T] \wedge Y \less X \approx_{C} 1_{N}
).
```

DEFINITION MunkTop.29.4: 2-ary function `Oneptcompactification`. If `TOPSP`$[X, T]$ then `Oneptcompactification`$(X, T) \simeq (! \langle Y, T' \rangle)($`COMPACTIFICATION`$[Y, T', X, T] \wedge$ $Y \backslash X \approx_{\mathtt{C}} 1_{\mathtt{N}})$.

2.3 Parsing

Let us review computer parsing methods and discuss the system that we use to parse LPT.

For the purposes of computer science, a *language* is nothing more than a set of finite strings over some alphabet Σ. The general problem of deciding membership of a string in a language is undecidable (just take the language consisting of all strings representing halting Turing-machine/input pairs, with respect to some encoding). But there is a large class of languages for which there are standard algorithms to decide membership: namely, the class of languages specified by what is known as a *context-free grammar*.

Languages in this class are actually more than just sets of strings; they have syntactic structure. And the algorithms that decide membership in these languages also determine the syntactic structure of the input. These algorithms are called *parsers*.

A context-free grammar or CFG is a special case of something more general, known as a *formal grammar*. A formal grammar defines a language by giving *production rules* by which all of the strings in the language can be "produced", starting from a single "start symbol". Formally, it is an ordered quadruple $G = (N, \Sigma, P, S)$, where N is called the set of *nonterminals*, or *nonterminal symbols*, Σ is called the *alphabet*, or the set of *terminals*, or *terminal symbols*, P is called the set of *productions*, and S is called the *start symbol*.

The sets N and Σ are disjoint. The start symbol S is an element of N. As for P, it is most easily described using standard computer science notation for sets of strings. If A and B are sets of symbols, then by A^* we denote the set of all finite strings of symbols taken from A (including the length zero "empty string" denoted e), and by AB we denote the set of all strings consisting of one element of A followed by one element of B. These notations may be used in combination. The set P of productions is then some set of ordered pairs contained in

$$(N \cup \Sigma)^* N (N \cup \Sigma)^* \times (N \cup \Sigma)^*,$$

and if $(\alpha, \beta) \in P$ we may express this by writing simply

$$\alpha \to \beta,$$

which is read, "α produces β".

In order to make use of this idea of "production" we define a binary relation $\Rightarrow$ on $(N \cup \Sigma)^*$ so that for all $\alpha, \beta \in (N \cup \Sigma)^*$, we have $\alpha \Rightarrow \beta$ just in case there are some

$\gamma, \delta, \varepsilon, \zeta$ in $(N \cup \Sigma)^*$ such that $\alpha = \gamma\delta\varepsilon$, $\beta = \gamma\zeta\varepsilon$, and $(\delta, \zeta) \in P$. We then define $\Rightarrow^*$ to be the transitive and reflexive closure of $\Rightarrow$.

The language $L(G)$ determined by a formal grammar $G = (N, \Sigma, P, S)$ is then defined to be the set of all α in Σ^* such that $S \Rightarrow^* \alpha$. Intuitively, you can reach the string α by starting from the start symbol S and using the productions in P.

Let us return now to the grammars that are of particular interest to us, the context-free grammars or CFG's. A CFG is any formal grammar in which the set P of productions is contained in

$$N \times (N \cup \Sigma)^*.$$

The idea is that now the productions simply tell you what a given nonterminal symbol $A \in N$ can produce; it is no longer possible to demand further that A be preceded and/or followed by some particular elements of $(N \cup \Sigma)^*$, i.e. that it be found in a certain "context". CFG's are the most important grammars in computer science because most important computer languages can be specified by them, and because there are efficient algorithms to parse the languages that they determine.

Of course, as one would expect, it is not as though we are designing parsing algorithms on a grammar-by-grammar basis, taking a given grammar G and then designing an algorithm to parse the language that it determines, then taking the next grammar G' and starting all over again. On the contrary, if $\mathcal{CFG}$ is the class of all context-free grammars then the general practice is to fix on some subclass $\mathcal{A} \subseteq \mathcal{CFG}$ and to design a parsing algorithm that will work for any grammar $G \in \mathcal{A}$.

This results in the development of computer programs called *compiler compilers*. You pass a formal grammar G belonging to the appropriate class $\mathcal{A}$ to a compiler compiler, and it compiles for you a compiler, which will parse the language of G. The reader may wonder why it is not called a "parser compiler" instead. This is only because a compiler is something more general than a parser, which involves a parser at its front end. A compiler compiler produces this more general kind of program, which one is free to use as though it were a mere parser.

Parsing algorithms that work for all of the grammars in $\mathcal{CFG}$ do exist; one popular case is the *Earley Algorithm*, named after J. Earley, who invented it in 1970. The reason that anyone bothers with algorithms that can handle only grammars belonging to proper subsets $\mathcal{A} \subset \mathcal{CFG}$ is that it is sometimes possible to design *faster* algorithms, if $\mathcal{A}$ is sufficiently restricted.

I have opted to use the Earley algorithm. Why not use something faster? A likely alternative would be the $LL(k)$ algorithms, which are commonly used in computer science; for example, for parsing programming languages.

By $LL(k)$ we mean an infinite class of algorithms. For any given positive integer

k the $LL(k)$ algorithm is designed to construct a left parse tree on the basis of no more than k input tokens beyond the current input pointer. The first 'L' stands for the left parse tree, and the other 'L' stands for the "lookahead" of at most k tokens. The runtime of the $LL(k)$ algorithm is linear in the length of the input [3], but only certain grammars – known as the LL grammars – can be parsed.

Whether LPT has any LL grammar is not known to me. On the other hand, I believe that the grammar that I have used for LPT is the grammar that one would most naturally write down; and this grammar is certainly not LL, as I prove in Appendix B. Briefly, the problem stems from a choice point in the grammar at which either a *term* or a *formula* could come next. (Consider the sort of formula given in point 8 in Section 2.1.1.) Either of these nonterminals can begin with an arbitrary number of parentheses, and this means the grammar is not $LL(k)$ for any k.

One response to this problem is to simply make k larger than the number of nested parentheses that any human being would ever want to look at. But lookahead comes at a cost in the $LL(k)$ algorithm. The size of the *parse table* used by the algorithm is in general, in the worst case, exponential in k [2].

Another solution would be to augment the basic $LL(k)$ algorithm. It is possible to parse arbitrarily nested parentheses, using a recursion stack [13]. But then we lose the advantage of a tried-and-true algorithm, as produced directly by a compiler compiler, on the basis of a formal grammar.

Yet another possible response would be to search for an alternative grammar for LPT, one which was LL. But this would probably require that we warp the grammar that we wanted to use quite a bit. There is an advantage to using a grammar which is immediately and intuitively correct to a human eye.

My response has been to simply use the Earley Algorithm to parse LPT. The algorithm is described in detail in [3]. The Earley algorithm can parse any context-free grammar, and it has runtime at worst $\mathcal{O}(n^3)$ (where n is the length of the input), and in fact runs in time $\mathcal{O}(n^2)$ if the grammar happens to be unambiguous. This is a runtime which could possibly become restrictive for long inputs, such as computer programs; but for input of the size of mathematical definitions it is no problem at all. In compiling my database of definitions the parser never ran for more than a couple of seconds.

I have used a compiler compiler known as ACCENT: "A Compiler Compiler for the ENTire class of context free grammars." It is described in [14]. One passes to ACCENT a description of the desired formal grammar, and it compiles a parser for this grammar that implements the Earley Algorithm.

Chapter 3

First application: Translation

As a first system utilizing the parsed LPT input, we have a translator program called `lpt2dzfc`, which will return for each input definition a formula of the language of DZFC ("Definitional ZFC"), to serve as a definitional axiom in that system.

As an example, consider the following definition, written in LPT, and taken from Appendix G:

```
DEFINITION MunkTop.12.1: 2-ary relation TOPOLOGY.
TOPOLOGY[\mathscr{T},X] \iff \mathscr{T} \subseteq \wp(X)
\wedge \varnothing \in \mathscr{T} \wedge X \in \mathscr{T}
\wedge (\forall S \subseteq \mathscr{T})
(\cup(S) \in \mathscr{T}) \wedge (\forall U,V \in \mathscr{T})
(U \cap V \in \mathscr{T}).
```

Above we have the text exactly as input. Our translator `lpt2dzfc` produces a LaTeX version of this input, still in LPT:

> DEFINITION MunkTop.12.1: 2-ary relation TOPOLOGY. $\texttt{TOPOLOGY}[\mathscr{T}, X] \leftrightarrow \mathscr{T} \subseteq \wp(X) \wedge \varnothing \in \mathscr{T} \wedge X \in \mathscr{T} \wedge (\forall S \subseteq \mathscr{T})(\cup(S) \in \mathscr{T}) \wedge (\forall U, V \in \mathscr{T})(U \cap V \in \mathscr{T})$.

as well as the translation of this definition into DZFC, also in LaTeX format:

> $\texttt{TOPOLOGY}[\mathscr{T}, X] \leftrightarrow \subseteq[\mathscr{T}, \wp(X)] \wedge \varnothing \in \mathscr{T} \wedge X \in \mathscr{T} \wedge (\forall S)(\subseteq[S, \mathscr{T}] \rightarrow \cup(S) \in \mathscr{T}) \wedge (\forall U, V)(U \in \mathscr{T} \wedge V \in \mathscr{T} \rightarrow \cap(U, V) \in \mathscr{T})$

Note that by translating into DZFC we lose qualified quantification. In the LPT version we have the clause,

$$(\forall S \subseteq \mathscr{T})(\cup(S) \in \mathscr{T}),$$

in which the variable S is not only quantified (by the universal quantifier), but also qualified, as being a subset of $\mathscr{T}$. In DZFC this is translated to

$$(\forall S)(\subseteq[S, \mathscr{T}] \rightarrow \cup(S) \in \mathscr{T}).$$

This is just one example of a feature of LPT that lends it its readability.

In the first section below I motivate the use of LPT as a language in which to compile a database of definitions, even when we have in mind the express purpose of using those definitions in formal verification systems, likely to operate in other languages.

In the subsequent sections I define the system DZFC and prove that it is a conservative extension of ZFC. Finally, in Section 3.5 I present sample definitions from Appendices F and G that showcase the special features of LPT.

3.1 Formalization, translation, and reduction

If we want to record a database of definitions, which language should we record them in? Let's assume that in the future we want to be able to easily write algorithms to translate these definitions into the languages of various formal systems, such as DZFC, or perhaps a definitional form of Peano Arithmetic. The problem then with many of the languages that we could choose from is that formalization in them requires a fair amount of *reduction*.

There is a standard ontology implicit in the informal language used in modern-day mathematics. Sets are very important objects, as are functions, and ordered tuples or sequences. Of course numbers persist as objects in informal mathematical talk, and there are various kinds of these: natural numbers, integers, rational numbers, real numbers, complex numbers, and more.

Meanwhile the official story in many formal systems is that there is only one kind of object, be they sets, functions, or natural numbers, and so some reduction must always take place when we formalize in these systems. If we formalize in ZFC we might reduce natural numbers to the von Neumann ordinals. If we formalize in a typed lambda calculus we might reduce sets to functions mapping from a given type to the type of truth values, thus reducing a set to its characteristic function. If we formalize in PA we might reduce finite sets to numbers, and reduce membership to a condition regarding the digits in the binary expansion of a number.

Formalization into any of the most popular formal systems will therefore be a two-step process: (1) We must identify the standard informal structures appearing in a given informal definition or theorem, i.e. we must identify the sets, functions,

tuples, numbers, etc. in terms of which the statement is given. Then (2) We must translate into the formal system, reducing the standard structures as necessary.

A language in which formalization requires a lot of reduction will not be a good language for our database of definitions, since reduction can be hard to reverse. This means that it would be hard to write algorithms to translate from the database language into various other languages.

So what we want is a language in which we can formalize informal definitions, and yet one which requires a minimum of reduction. It will be easy later on to set up mechanical translations from this language into others. Clearly, a language that requires little reduction will just be one that features locutions to accommodate talk of as many as possible of the sorts of objects that belong to the standard informal ontology of modern-day mathematical discourse. This is just what LPT provides. It gives ways to denote sets, functions, and tuples.

But why not just record definitions in natural language? Well, as of now good algorithms for natural language processing are still very experimental. But that's not really the important point anyway. We may very well have good algorithms soon. More importantly, I anticipate that if we had a system to translate directly from natural language into, say, the language of DZFC, it would probably work by passing through a stage in which it had refined an informal definition into something a lot like what we get when we translate into LPT by hand. Standard techniques of formalization work by fixing on the standard structures in the informal ontology and then reducing these. We will want to continue using these techniques, and so we will continue to want to translate informal definitions into standard formal systems by first noting which of these structures occur in them, and then formalizing in terms of these. If we are going to pass through a representation like what we get in LPT anyway, then we are not losing anything (at least not much) by recording our definitions directly in LPT.

So much is true at least for mathematical text written expressly for computer storage. Meanwhile, we may also wish to produce electronic versions of existing texts, for MKM purposes. In that case, a natural language parser would indeed be desirable.

3.2 A translation target system

We hope that LPT will be amenable to translation into most any popular formal system, since formalization in LPT should require little reduction. On the other hand, many of the definitions that we may encounter in textbooks, at least those

given in a foundational development of mathematics, will already involve quite a bit of reduction, of one kind or another. If we are following a set-theoretic development of the number systems from $\mathbb{N}$ on up to $\mathbb{C}$ for example, then the definitions will involve reductions that we would not want to use if, say, we were developing mathematics in arithmetic instead of in set theory. In this way our choice of definitions to formalize does represent some loss of the general applicability of LPT.

In the definitions that I compile in Appendices F and G I follow a set-theoretic development. Accordingly, we need a computer program to translate from LPT into some system of set theory. ZFC would be the most obvious choice, but in fact a system that offers a little bit more will work much better for formalizers wishing to work (ultimately) in ZFC.

Even an informal proof of modest length or complexity can be essentially unintelligible to a human reader if fully formalized in ZFC. We lose the advantages of defined functions and relations if we try to work directly in ZFC. In order to formalize a given proof that uses defined functions $f_1, \ldots, f_n$ and defined relations $R_1, \ldots, R_m$, it makes sense to work in a definitional extension of ZFC that features function and relation strings to stand for the f_i and the R_j, and definitional axioms to allow formal deduction with these functions and relations. The definitional axioms for the defined functions in turn demand that we provide a description operator.

These considerations motivate the design of a program to translate from LPT to DZFC, rather than to ZFC itself. My program `lpt2dzfc` serves this purpose. It is impractical to set up a further translation, from DZFC to ZFC, since the former gives an iterated exponential speedup over the latter. This means that it is impossible to bound the increase in length by any fixed stack of exponentials [5]. Fortunately however, DZFC is a conservative extension of ZFC, as I prove in the sections below. This means that formalizers wishing to ground theorems in the axioms of ZFC may instead work in DZFC, and simply invoke conservativity.

3.3 DZFC

To get DZFC we just add to ZFC those features needed to work with defined functions and relations. Primarily this means the expansion of the language to include infinite collections of function and relation strings, as well as the inclusion of a description operator, ι.

We also design DZFC to use *free logic*, i.e., logic in which terms may fail to denote. In this we follow Feferman [8]. This allows us to make function definitions more natural, by leaving functions undefined at certain arguments. For example, in

developing arithmetic we need not define division by zero. In order to use free logic we include in DZFC a definedness predicate $\downarrow$, and a partial equality relation $\simeq$.

DZFC is our basic definitional system. If σ is a set of function and relation symbols then we write DZFC(σ) for the system obtained by adding to DZFC a definitional axiom for each symbol in σ.

Given a set $\sigma = \{f_1, \ldots, f_n, R_1, \ldots, R_m\}$ of defined function and relation symbols, the axioms and rules of inference for DZFC(σ) are as follows.

I. *Propositional Axioms and Rules*

(i) Axioms: All substitution instances of tautologies.

(ii) Rules:

$$\text{MP} \qquad \frac{A \to B \quad A}{B}$$

II. *Quantificational Axioms and Rules*

(i) Axioms:

$$\forall - \text{Ax} \qquad \forall x A(x) \wedge (t\downarrow) \to A(t)$$

$$\exists - \text{Ax} \qquad A(t) \wedge (t\downarrow) \to \exists x A(x)$$

where t is free for x in A.

(ii) Rules:

$$\forall\text{I} \qquad \frac{B \to A(x)}{B \to \forall x A(x)}$$

$$\exists\text{E} \qquad \frac{A(x) \to B}{\exists x A(x) \to B}$$

where '$\mathtt{x}'$ is not free in B.

III. *Equality and Substitution Axioms*

(i) $x = x$

(ii) $s = t \to t = s$

(iii) $r = s \wedge s = t \to r = t$

(iv) $s_1 \in s_2 \wedge \vec{s} = \vec{t} \to t_1 \in t_2$

(v) $(s\downarrow) \wedge s = t \to (t\downarrow)$

(vi) $s \simeq t \leftrightarrow ((s\downarrow) \vee (t\downarrow) \rightarrow s = t)$

(vii) $R_j(\vec{s}) \wedge \vec{s} = \vec{t} \rightarrow R_j(\vec{t})$

(viii) $\vec{s} = \vec{t} \rightarrow f_i(\vec{s}) \simeq f_i(\vec{t})$

IV. *Set Axioms.* The usual axioms of ZFC, where the formulas in the Separation and Replacement schemata range over all formulas of DZFC.

V. *Description Axiom*

$$\text{DES} \qquad y = (\iota x)A(x) \leftrightarrow \forall z(A(z) \leftrightarrow z = y).$$

VI. *Definedness Axioms*

(i) $x\downarrow$

(ii) $\varnothing\downarrow$

(iii) $s = t \rightarrow (s\downarrow) \wedge (t\downarrow)$

(iv) $s \in t \rightarrow (s\downarrow) \wedge (t\downarrow)$

(v) $f_i\downarrow$ for nullary functions f_i

(vi) $f_i(\vec{t})\downarrow \rightarrow (\vec{t}\downarrow)$

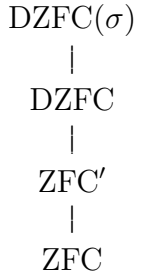
(vii) $R_j(\vec{t}) \rightarrow (\vec{t}\downarrow)$

VII. *Definitional Axioms.* For each function symbol f_i we add a definitional axiom of the form $f_i\vec{x} \simeq (\iota y)A_{f_i}(\vec{x}, y)$, where f_k does not appear in the formula A_{f_i} for $k \geq i$. Likewise, for each relation symbol R_j we add a definitional axiom of the form $R_j(\vec{x}) \leftrightarrow \phi_{R_j}(\vec{x})$, where R_k does not appear in the formula ϕ_{R_j} for $k \geq j$.

$$\begin{aligned} f_1\vec{x} &\simeq (\iota y)A_{f_1}(\vec{x}, y) \\ &\vdots \\ f_n\vec{x} &\simeq (\iota y)A_{f_n}(\vec{x}, y) \\ R_1(\vec{x}) &\leftrightarrow \phi_{R_1}(\vec{x}) \\ &\vdots \\ R_m(\vec{x}) &\leftrightarrow \phi_{R_m}(\vec{x}) \end{aligned}$$

3.4 Conservativity

We prove conservativity of DZFC(σ) over ZFC, for σ an arbitrary set of defined functions and relations. We go in stages, first defining an intermediate system ZFC$'$, and then proving three conservative extension theorems, yielding the hierarchy depicted below:



where each system is a conservative extension of the system below it. Since conservativity is transitive we get our desired result.

DZFC has a *partial* description operator, meaning that terms of the form $(\iota x)A(x)$ do not always denote. Namely, such a term fails to denote just in case there does not exist a unique x such that $A(x)$. In fact the only way in which a term in DZFC can fail to denote is if it contains such a description, and the definedness predicate $\downarrow$ and the partial equality relation $\simeq$ are included in DZFC in order to accommodate these non-denoting terms.

In the intermediate system ZFC$'$ we extend ZFC by introducing a *total* description operator. In ZFC$'$ terms of the form $(\iota x)A(x)$ always denote, and there is no need for $\downarrow$ or $\simeq$. The idea is that the axiom for the description operator includes a "default" clause, so that if there does not exist a unique x such that $A(x)$ then $(\iota x)A(x)$ simply denotes a default value, for which we have chosen the empty set.

3.4.1 The four systems

The axioms and rules of inference for the four systems are as follows (we present DZFC(σ) again for convenience).

1. ZFC.

   (I) *Propositional Axioms and Rules*

   i. Axioms: All substitution instances of tautologies.

ii. Rules:

$$\text{MP} \qquad \frac{A \to B \quad A}{B}$$

(II) *Quantificational Axioms and Rules*

i. Axioms:

$$\forall - \text{Ax} \qquad \forall x A(x) \to A(t)$$

$$\exists - \text{Ax} \qquad A(t) \to \exists x A(x)$$

where t is free for x in A.

ii. Rules:

$$\forall \text{I} \qquad \frac{B \to A(x)}{B \to \forall x A(x)}$$

$$\exists \text{E} \qquad \frac{A(x) \to B}{\exists x A(x) \to B}$$

where '`x'` is not free in B.

(III) *Equality and Substitution Axioms*

i. $x = x$

ii. $s = t \to t = s$

iii. $r = s \wedge s = t \to r = t$

iv. $s_1 \in s_2 \wedge \vec{s} = \vec{t} \to t_1 \in t_2$

(IV) *Set Axioms*

i. Axiom of Extensionality:

$$\forall X \forall Y \forall u (u \in X \leftrightarrow u \in Y) \to X = Y$$

ii. Axiom of the Unordered Pair:

$$\forall a \forall b \exists c \forall x (x \in c \leftrightarrow (x = a \vee x = b))$$

iii. Schema of Separation:

$$\forall X \forall p \exists Y \forall u (u \in Y \leftrightarrow (u \in X \wedge \varphi(u, p)))$$

iv. Axiom of Union:

$$\forall X \exists Y \forall u (u \in Y \leftrightarrow \exists z (z \in X \wedge u \in z))$$

v. Axiom of the Power Set:

$$\forall X \exists Y \forall u (u \in Y \leftrightarrow \forall z (z \in u \rightarrow z \in X))$$

vi. Axiom of Infinity:

$$\exists S (\varnothing \in S \wedge (\forall x \in S)(x \cup \{x\} \in S))$$

vii. Schema of Replacement:

$$\forall p \, [\forall x \forall y \forall z (\varphi(x, y, p) = \varphi(x, z, p) \rightarrow y = z)$$

$$\rightarrow \forall X \exists Y \forall y (y \in Y \leftrightarrow (\exists x \in X)(\varphi(x, y, p)))]$$

viii. Axiom of Foundation:

$$\forall S (S \neq \varnothing \rightarrow (\exists x \in S)(S \cap x = \varnothing))$$

ix. Axiom of Choice:

$$\forall A \, [\forall x \forall y (x \in A \wedge y \in A \rightarrow x \cap y = \varnothing \wedge x \neq \varnothing)$$

$$\rightarrow \exists C \forall x (x \in A \rightarrow \exists y \forall z (z \in x \cap C \leftrightarrow z = y)))]$$

2. ZFC′. To the language we add the symbol ι, a description operator. It will be axiomatized so that it is total, not partial; thus, $(\iota x)A(x)$ is always defined, for any formula A. The idea is that if in fact there is a unique x such that $A(x)$ then $(\iota x)A(x)$ will return this very x; otherwise it will return a default value, namely, the empty set $\varnothing$. The axioms and rules are as follows.

(I) *Propositional Axioms and Rules.* Same as ZFC.

(II) *Quantificational Axioms and Rules.* Same as ZFC.

(III) *Equality and Substitution Axioms.* Same as ZFC.

(IV) *Set Axioms.* Expand the separation and replacement schemata so that φ is any formula of the new language.

(V) *Description Axiom*

$$\text{t} - \text{DES} \qquad y = (\iota x)A(x) \leftrightarrow [\forall z (A(z) \leftrightarrow z = y) \vee (\neg \exists ! z A(z) \wedge y = \varnothing)].$$

3. DZFC. The language is the same as for ZFC′, except that we add a definedness predicate $\downarrow$, and a partial equality symbol $\simeq$. The description axiom is changed so that now ι will be a partial description operator instead of total. This means that we remove the "defaulting mechanism", so that now $(\iota x)A(x)$ is simply "undefined" if there is not a unique x such that $A(x)$.

   What it really means to say that now some terms will be "undefined" amounts primarily to a change in the quantificational axioms: they become more stringent, their antecedents now requiring that the terms involved be "defined". Here, a term is "defined" just in case it satisfies the new definedness predicate $\downarrow$. Thus we also need new axioms with which to infer this definedness.

   Axioms and rules are as follows.

   (I) *Propositional Axioms and Rules.* Same as ZFC′.

   (II) *Quantificational Axioms and Rules.* Rules the same as ZFC′. The axioms become:

$$\forall - \text{Ax} \qquad \forall x A(x) \wedge (t\downarrow) \rightarrow A(t)$$

$$\exists - \text{Ax} \qquad A(t) \wedge (t\downarrow) \rightarrow \exists x A(x)$$

   where t is free for x in A.

   (III) *Equality and Substitution Axioms.* Those of ZFC′, plus:

$$(s\downarrow) \wedge s = t \rightarrow (t\downarrow)$$

$$s \simeq t \leftrightarrow ((s\downarrow) \vee (t\downarrow) \rightarrow s = t)$$

   (IV) *Set Axioms.* Expand the separation and replacement schemata so that φ is any formula of the new language.

   (V) *Description Axiom*

$$\text{p} - \text{DES} \qquad y = (\iota x)A(x) \leftrightarrow \forall z(A(z) \leftrightarrow z = y).$$

   (VI) *Definedness Axioms*

   i. $x\downarrow$
   ii. $\varnothing\downarrow$
   iii. $s = t \rightarrow (s\downarrow) \wedge (t\downarrow)$
   iv. $s \in t \rightarrow (s\downarrow) \wedge (t\downarrow)$

4. DZFC(σ). We have a set $\sigma = \{f_1, \ldots, f_n, R_1, \ldots, R_m\}$ of function and relation symbols to be defined. For each function symbol f_i we add a definitional axiom of the form $f_i\vec{x} \simeq (\iota y)A_{f_i}(\vec{x}, y)$, where f_k does not appear in the formula A_{f_i} for $k \geq i$. For each relation symbol R_j we add a definitional axiom of the form $R_j(\vec{x}) \leftrightarrow \phi_{R_j}(\vec{x})$, where R_k does not appear in the formula ϕ_{R_j} for $k \geq j$.

   (I) *Propositional Axioms and Rules.* Same as DZFC.

   (II) *Quantificational Axioms and Rules.* Same as DZFC.

   (III) *Equality and Substitution Axioms.* Those of DZFC, plus:

$$R_j(\vec{s}) \wedge \vec{s} = \vec{t} \rightarrow R_j(\vec{t})$$

$$\vec{s} = \vec{t} \rightarrow f_i(\vec{s}) \simeq f_i(\vec{t})$$

   (IV) *Set Axioms.* Same as DZFC.

   (V) *Description Axiom.* Same as DZFC.

   (VI) *Definedness Axioms.* Those of DZFC, plus:

$$f_i\!\downarrow, \text{for nullary functions} f_i$$

$$f_i(\vec{t})\!\downarrow \rightarrow (\vec{t}\!\downarrow)$$
$$R_j(\vec{t}) \rightarrow (\vec{t}\!\downarrow)$$

   (VII) *Definitional Axioms.* As described above,

$$\begin{array}{rcl} f_1\vec{x} & \simeq & (\iota y)A_{f_1}(\vec{x}, y) \\ & \vdots & \\ f_n\vec{x} & \simeq & (\iota y)A_{f_n}(\vec{x}, y) \\ R_1(\vec{x}) & \leftrightarrow & \phi_{R_1}(\vec{x}) \\ & \vdots & \\ R_m(\vec{x}) & \leftrightarrow & \phi_{R_m}(\vec{x}) \end{array}$$

3.4.2 The extension theorems

Our first theorem states that ZFC' is a conservative extension of ZFC. This is the first among three conservative extension theorems that we have to prove, and the basic strategy will be the same each time.

To show that a formal theory T' is a conservative extension of a subsystem T we will define a translation * from the formulas of T' to the formulas of T in such a way that whenever a formula A of T' is also a formula of the subsystem T then A^* is provably equivalent to A in the system T. Then we just need to prove that for every theorem A of T', A^* is a theorem of T.

We use induction on the length of the shortest proof of A in T'. Since the shortest any proof can be is one line, and since the only formulas with one-line proofs are axioms, this means that the base step consists precisely of showing that for every *axiom* A of T', A^* is a theorem of T.

The induction step is easy, and we get it out of the way right now. Suppose A is a theorem of T' whose shortest proof is n lines. Then there is some rule of inference R in T', and some set $N \subseteq \{1, 2, \ldots, n-1\}$ such that A can be inferred in T' by applying rule R to the lines of the proof whose line numbers are in N. Let us denote the formula on line k by B_k. Then for each $k \in N$, B_k is a theorem of T', and therefore, by the inductive hypothesis, B_k^* is a theorem of T. It would suffice then to show that A^* follows in T from $\{B_k^*\}_{k \in N}$.

Thus, for the induction step it suffices to show that for all rules of inference R of T', all sets $\mathcal{H}$ of hypotheses, and all sentences A, we have

$$\frac{\mathcal{H}}{A}\ R \quad \Rightarrow \quad \mathcal{H}^* \vdash_T A^*. \tag{3.1}$$

All four of our logical systems have the same three rules of inference: the one propositional rule of *modus ponens*, and the two quantificational rules of *universal introduction* and *existential elimination*. Therefore (3.1) holds provided the * translation meets two criteria:

i. * commutes with $\rightarrow$ and with the quantifiers $\forall$ and $\exists$, and

ii. if * introduces any free variables they are always fresh.

In fact every translation we define below will meet these two criteria, so the induction step is done for each of our conservative extension theorems. In each case we now need only show the base step, that each axiom of T' translates to a theorem of T.

We return now to our first goal, of proving that ZFC' is a conservative extension of ZFC. We define a translation, and prove in several lemmas that the axioms of ZFC' translate to theorems of ZFC.

The translation * from ZFC' to ZFC will make use of two auxiliary functions, one the "underscore" function, and the other a function called Δ. We define all three

simultaneously. All of these techniques and notations come from [16], Chapter 2, Section 7.

The Δ function will be defined on terms t, and for each term t will return a formula $\Delta(t)$. We will extend the domain of Δ to the set of all finite sequences $\vec{t}$ of terms by the convention that

$$\Delta(\vec{t}) = \bigwedge_{i=1}^{|\vec{t}|} \Delta(t_i).$$

In this and later sections we will use other functions similar to Δ and will use the same convention for these, without comment.

We also use an indexing of the occurrences of the description operator ι in formulas. Thus, we write $(\iota y)B(y)$ instead as $(\iota_k y)B(y)$ with k an index. All indices are distinct, and beyond this constraint our proof is independent of the manner of indexing.

For each occurrence ι_k of the description operator we will be introducing a variable, assumed to be fresh, denoted y_k. This correspondence will be implicit when we introduce a whole sequence $\vec{y}$ of variables corresponding to a formula containing several occurrences of the description operator.

The idea behind the translation * is simple. In his theory of definite descriptions Russell proposed that any statement of the form

$$P((\iota x)A(x))$$

is equivalent to the statement

$$\exists x(P(x) \wedge \forall y(A(y) \leftrightarrow y = x)).$$

Our translation * merely uses this idea in order to remove occurrences of the description operator. (Of course this Russellian idea is the same one behind our Description Axioms, and this is why the translation works.)

With P an atomic formula, and B an arbitrary formula, we define * and its two

auxiliary functions as follows:

* commutes with $\wedge$, $\vee$, $\neg$, $\rightarrow$, $\forall$, $\exists$;
$P(\vec{t})^* := \exists \vec{y} \,.\, P(\underline{\vec{t}}) \wedge \Delta(\vec{t})$;
$\top^* := \top$;
$\bot^* := \bot$;
$\underline{x} := x$;
$\underline{\varnothing} := \varnothing$;
$\underline{(\iota_k y)B(y)} := y_k$;
$\overline{\Delta(x) := \top}$;
$\Delta(\varnothing) := \top$;
$\Delta((\iota_k z)B(\vec{t}, z)) := [(\exists! z B(\vec{t}, z)^*) \wedge B(\vec{t}, y_k)^*] \vee [\neg(\exists! z B(\vec{t}, z)^*) \wedge y_k = \varnothing]$.

For the remainder of this subsection, we write T' to mean ZFC', and we write T to mean ZFC.

Now we show for each axiom A of T' that $\vdash_T A^*$. For the propositional axioms this is obvious.

For the set theoretic axioms it is also obvious. All but the separation and replacement schemata translate to formulas obviously provably equivalent to themselves. And when translating an axiom belonging to either of the two schemata we need only note that the formula φ of T' appearing there translates to some formula of T to see that the axiom translates to a member of the corresponding schema in T.

This leaves (1) the description axiom, (2) the equality and substitution axioms, and (3) the quantificational axioms. We handle these in order. (Note that the reason we must check items (2) and (3) is that these axioms involve terms, which vary over a wider range in T' than they do in T.)

Notation: We use '$=$' when formulas are the same, and '$\Leftrightarrow$' when formulas are provably equivalent in T.

First the description axiom:

Lemma 1 *For all formulas $A(x)$ of T',*

$$\vdash_T (y = (\iota x)A(x) \leftrightarrow [(\exists! x A(x) \wedge A(y)) \vee (\neg\exists! x A(x) \wedge y = \varnothing)])^*.$$

Proof.. We want to see that

$$\vdash_T (y = (\iota x)A(x))^* \leftrightarrow [(\exists! x A(x)^* \wedge A(y)^*) \vee (\neg\exists! x A(x)^* \wedge y = \varnothing)].$$

But this is obvious once we observe that

$$(y = (\iota x)A(x))^* = \exists y' \,.\, [(\exists! x A(x)^* \wedge A(y')^*) \vee (\neg\exists! x A(x)^* \wedge y' = \varnothing)] \wedge y = y'.$$

□

Next we consider the four equality and substitution axioms. For the first of these, the desired result is immediate. We will examine the second. The third and fourth are similar.

Lemma 2 *For any terms s and t of T', we have $\vdash_T (s = t \to t = s)^*$.*

Proof. If both s and t are variables or constants the result is trivial. Supposing $s = (\iota z)B(z)$ and t is a variable or a constant, we have

$$(s = t)^* \Leftrightarrow \exists y \,.\, y = t \wedge \Delta((\iota z B(z))$$

and

$$(t = s)^* \Leftrightarrow \exists y \,.\, t = y \wedge \Delta((\iota z B(z))$$

and clearly T proves that the one implies the other.

Finally, supposing $s = (\iota z)B(z)$, and $t = (\iota z)C(z)$ we have

$$(s = t)^* \Leftrightarrow \exists y_1, y_2 \,.\, y_1 = y_2 \wedge \Delta((\iota z B(z)) \wedge \Delta((\iota z C(z))$$

and

$$(t = s)^* \Leftrightarrow \exists y_1, y_2 \,.\, y_2 = y_1 \wedge \Delta((\iota z B(z)) \wedge \Delta((\iota z C(z))$$

and once again the result is clear. □

Finally we must show that the two quantificational axioms of T' translate to theorems of T. In order to do this we define a second translation, †. This translation will make use of the same Russellian theory of definite descriptions, adding clauses to deal with existence and uniqueness. The only difference is that whereas the * translation kept these extra clauses conjoined right beside the atomic formulas, the † translation will put these clauses on the outside of the whole formula. In the next lemma we establish that for any formula in T', its two translations into T, under * and under †, are provably equivalent in T.

For the † translation we use an auxiliary function Γ slightly different from Δ (notice that there are no *'s on the right-hand side in its definition below); we use the same underscore function. For an arbitrary formula A we write $A(\vec{t})$ to mean that $\vec{t}$ is the sequence of all distinct terms appearing anywhere in the formula A, in order from left to right. A bound variable is not considered to be a term. So, for example, if we had

$$A \equiv (x \in y) \wedge (x \notin \varnothing) \wedge (y = (\iota z)B(\vec{w}, z)) \wedge \exists u(u = \varnothing)$$

then we would write

$$A \equiv A(x, y, \varnothing, (\iota z)B(\vec{w}, z), \vec{w}).$$

Let A and B be arbitrary formulas. The definition of † is as follows.

$A(\vec{t})^\dagger := \exists \vec{y}\,.\, A(\underline{\vec{t}}) \wedge \Gamma(\vec{t})$;
$\top^\dagger := \top$;
$\bot^\dagger := \bot$;
$\underline{x} := x$;
$\underline{\varnothing} := \varnothing$;
$\underline{(\iota_k y)B(y)} := y_k$;
$\underline{\Gamma(x)} := \top$;
$\Gamma(\varnothing) := \top$;
$\Gamma((\iota_k z)B(\vec{t}, z)) := [(\exists! z B(\underline{\vec{t}}, z)) \wedge B(\underline{\vec{t}}, y_k)] \vee [\neg(\exists! z B(\underline{\vec{t}}, z)) \wedge y_k = \varnothing]$.

Lemma 3 *For every formula A of T', we have*

$$\vdash_T A^* \leftrightarrow A^\dagger. \tag{3.2}$$

Proof. Again following [16], Chapter 2, Section 7, we begin by defining a "degree" δ on the formulas and terms of T' as follows. With B and C any formulas, and P an atomic formula, we define:

$\delta(x) := 0$;
$\delta(\varnothing) := 0$;
$\delta((\iota z)B(z)) := 1 + \delta(B(z))$;
$\delta(P(\vec{t})) := \delta(\vec{t}) := \max(\delta(t_1), \ldots, \delta(t_n))$;
$\delta(\bot) := 0$;
$\delta(\top) := 0$;
$\delta(\neg B) := 1 + \delta(B)$;
$\delta(B \circ C) := 1 + \max(\delta(B), \delta(C))$ for $\circ \in \{\wedge, \vee, \rightarrow\}$;
$\delta(\forall x B) = \delta(\exists x B) := 1 + \delta(B)$.

The proof is by complete induction on $\delta(A)$.

Base step: $\delta(A) = 0$. Then either $A = \top$, $A = \bot$, or $A = P(x, \vec{v})$ for some variables $\vec{v} = v_1, \ldots, v_n$ distinct from x and some atomic formula P. In all of these cases the result is obvious.

Induction step: $\delta(A) = n + 1$. There are seven cases to consider:

i. $A = B \vee C$,

ii. $A = B \wedge C$,

iii. $A = B \to C$,

iv. $A = \neg B$,

v. $A = \exists z B$,

vi. $A = \forall z B$, and

vii. $A = P(\vec{t})$.

In the first six cases we use the induction hypothesis and the fact that * commutes with the logical connectives and quantifiers to reduce to the goal of showing that

i. $\vdash_L A^\dagger \leftrightarrow B^\dagger \vee C^\dagger$,

ii. $\vdash_L A^\dagger \leftrightarrow B^\dagger \wedge C^\dagger$,

iii. $\vdash_L A^\dagger \leftrightarrow B^\dagger \to C^\dagger$,

iv. $\vdash_L A^\dagger \leftrightarrow \neg B^\dagger$,

v. $\vdash_L A^\dagger \leftrightarrow \exists z B^\dagger$, and

vi. $\vdash_L A^\dagger \leftrightarrow \forall z B^\dagger$.

All six cases are easy. Finally, case (vii), in which A is atomic, is obvious. □

At last we want to show that for every quantificational axiom A of T', we have $\vdash_T A^*$. But by lemma 3.2 it is enough to show that $\vdash_T A^\dagger$. Therefore we will prove the following lemma.

Lemma 4 *For all formulas A of T' and terms t of T' such that t is free for x in A, we have*

$$\vdash_T \forall x A^\dagger \to ([t/x]A)^\dagger \tag{3.3}$$

and

$$\vdash_T ([t/x]A)^\dagger \to \exists x A^\dagger. \tag{3.4}$$

Proof. Since $([t/x]A)^\dagger$ is provably equivalent in T to $\exists y \,.\, [y/x]A^\dagger \wedge \Gamma(t)$, (3.4) follows immediately. To get (3.3) we use a case analysis in T on $(\exists! z C(z)) \vee \neg(\exists! z C(z))$, where $t = (\iota z)C(z)$. □

This completes our first conservative extension theorem.

Theorem 5 ZFC$'$ *is a conservative extension of ZFC.*

3.4.3 Second extension

Next we prove that DZFC is a conservative extension of ZFC′. The overall structure of the proof is the same as in the last section. This time we define the following translation from DZFC to ZFC′.

$$
\begin{array}{l}
^\dagger \text{ commutes with } \wedge,\ \vee,\ \neg,\ \rightarrow,\ \forall,\ \exists;\\
P(\vec{t})^\dagger := P(\underline{\vec{t}}) \wedge (\vec{t}\downarrow)^\dagger, \text{for} P(\vec{t}) \not\equiv (\vec{t}\downarrow);\\
(x\downarrow)^\dagger := \top;\\
((\iota x)B(x)\downarrow)^\dagger := \exists! x B(x)^\dagger;\\
\underline{x} := x;\\
\underline{(\iota x)B(x)} := (\iota x)B(x)^\dagger.
\end{array}
$$

Now we check that the axioms of DZFC translate to provable formulas of ZFC′. As in the last section, the propositional axioms and the set axioms are obvious.

We write T^* for DZFC, and T' for ZFC′.

Lemma 6 *The translation of the description axiom of T^* is provable in T'.*

Proof. The translation of $(\iota x)A(x) = y \leftrightarrow \forall x[A(x) \leftrightarrow x = y]$ is provably equivalent to

$$(\iota x)A(x)^\dagger = y \wedge \exists! x A(x)^\dagger \leftrightarrow \forall x[A(x)^\dagger \leftrightarrow x = y].$$

And, given the description axiom of T', this last is provably equivalent in T' to

$$[\forall z(A(z)^\dagger \leftrightarrow z = y) \vee (\neg\exists! z A(z)^\dagger \wedge y = \varnothing)] \wedge \exists! x A(x)^\dagger \leftrightarrow \forall x[A(x)^\dagger \leftrightarrow x = y],$$

which clearly T' proves. □

We turn now to the equality axioms.

Lemma 7 *For all terms s, t of T^*, we have $T' \vdash (s = t \rightarrow t = s)^\dagger$.*

Proof. Whatever s and t are, we have

$$(s = t \rightarrow t = s)^\dagger \equiv \underline{s} = \underline{t} \wedge (s\downarrow)^\dagger \wedge (t\downarrow)^\dagger \rightarrow \underline{t} = \underline{s} \wedge (t\downarrow)^\dagger \wedge (s\downarrow)^\dagger$$

and clearly T' proves the left-hand side. □

Lemma 8 *For all terms r, s, t of T^*, we have $T' \vdash (r = s \wedge s = t \rightarrow r = t)^\dagger$.*

Proof is similar.

Next the substitution axioms.

Lemma 9 $T' \vdash (R(\vec{s}) \wedge \vec{s} = \vec{t} \rightarrow R(\vec{t}))^\dagger$.

Proof.

$$(R(\vec{s}) \wedge \vec{s} = \vec{t} \rightarrow R(\vec{t}))^\dagger \Leftrightarrow R(\underline{\vec{s}}) \wedge (\vec{s}\downarrow)^\dagger \wedge \underline{\vec{s}} = \underline{\vec{t}} \wedge (\vec{t}\downarrow)^\dagger \rightarrow R(\underline{\vec{t}}) \wedge (\vec{t}\downarrow)^\dagger$$

and the left-hand side follows from substitution in T'. □

It is obvious from our choice of translation that the definedness axioms translate to provable formulas in T'.

Finally we come to the quantificational axioms. First we prove a supporting lemma (which makes up the bulk of this section).

Lemma 10 *For all formulas A and terms t of T^*, if x is free in A and t is free for x in A, then $T' \vdash [\underline{t}/x](A^\dagger) \wedge (t\downarrow)^\dagger \rightarrow ([t/x]A)^\dagger$.*

Proof. It suffices to prove the lemma under the additional assumption that A is in negation-normal form ($\neg$ appears only immediately before atomic formulae), and does not contain $\rightarrow$. Thus we begin by defining a degree δ on the formulas and terms of T^* as follows:

$$\begin{array}{l}
\delta(x) := 0; \\
\delta((\iota x)B(x)) := 1 + \delta(B(x)); \\
\delta(P(\vec{t})) := \delta(\vec{t}); \\
\delta(\neg B) := \delta(B); \\
\delta(B \circ C) := 1 + \max(\delta(B), \delta(C)), \text{for} \circ \in \{\wedge, \vee\}; \\
\delta(\forall x B) = \delta(\exists x B) := 1 + \delta(B).
\end{array}$$

The proof is by complete induction on $\delta(A)$.

Base step: $\delta(A) = 0$. There are two cases to consider:

i. $A \equiv P(x, \vec{v})$, and

ii. $A \equiv \neg P(x, \vec{v})$.

We enumerate them below.

i. First we compute that

$$\begin{aligned}
[\underline{t}/x](A^\dagger) &\equiv [\underline{t}/x](P(x,\vec{v})^\dagger) \\
&\equiv [\underline{t}/x](P(\underline{x},\underline{\vec{v}}) \wedge (x\downarrow)^\dagger \wedge (\vec{v}\downarrow)^\dagger) \\
&\Leftrightarrow [\underline{t}/x]P(x,\vec{v}) \\
&\equiv P(\underline{t},\vec{v})
\end{aligned}$$

and then that

$$\begin{aligned}
([t/x]A)^\dagger &\equiv ([t/x]P(x,\vec{v}))^\dagger \\
&\equiv P(t,\vec{v})^\dagger \\
&\equiv P(\underline{t},\underline{\vec{v}}) \wedge (t\downarrow)^\dagger \wedge (\vec{v}\downarrow)^\dagger \\
&\Leftrightarrow P(\underline{t},\vec{v}) \wedge (t\downarrow)^\dagger
\end{aligned}$$

Thus $T' \vdash [\underline{t}/x](A^\dagger) \wedge (t\downarrow)^\dagger \rightarrow ([t/x]A)^\dagger$ is just

$$T' \vdash P(\underline{t},\vec{v}) \wedge (t\downarrow)^\dagger \rightarrow P(\underline{t},\vec{v}) \wedge (t\downarrow)^\dagger.$$

ii. Now we have $A \equiv \neg P(x,\vec{v})$. This time the $(t\downarrow)^\dagger$ conjunct is really not needed, as we find:

$$\begin{aligned}
[\underline{t}/x](A^\dagger) &\equiv [\underline{t}/x](\neg P(x,\vec{v})^\dagger) \\
&\equiv [\underline{t}/x]\neg(P(\underline{x},\underline{\vec{v}}) \wedge (x\downarrow)^\dagger \wedge (\vec{v}\downarrow)^\dagger) \\
&\Leftrightarrow [\underline{t}/x]\neg P(x,\vec{v}) \\
&\equiv \neg P(\underline{t},\vec{v}),
\end{aligned}$$

while

$$\begin{aligned}
([t/x]A)^\dagger &\equiv ([t/x]\neg P(x,\vec{v}))^\dagger \\
&\equiv \neg P(t,\vec{v})^\dagger \\
&\equiv \neg(P(\underline{t},\underline{\vec{v}}) \wedge (t\downarrow)^\dagger \wedge (\vec{v}\downarrow)^\dagger) \\
&\Leftrightarrow \neg P(\underline{t},\vec{v}) \vee \neg(t\downarrow)^\dagger.
\end{aligned}$$

Of course $T' \vdash \neg P(\underline{t},\vec{v}) \rightarrow \neg P(\underline{t},\vec{v})$. And we can strengthen the antecedent and weaken the consequent to get

$$T' \vdash \neg P(\underline{t},\vec{v}) \wedge (t\downarrow)^\dagger \rightarrow \neg P(\underline{t},\vec{v}) \vee \neg(t\downarrow)^\dagger$$

which is $T' \vdash [\underline{t}/x](A^\dagger) \wedge (t\downarrow)^\dagger \rightarrow ([t/x]A)^\dagger$ in this case.

Induction step: $\delta(A) = n + 1$. There are six cases to consider:

i. $A \equiv B \vee C$,

ii. $A \equiv B \wedge C$,

iii. $A \equiv \exists z B$,

iv. $A \equiv \forall z B$,

v. $A \equiv P(\vec{s})$, and

vi. $A \equiv \neg P(\vec{s})$.

We enumerate them below. The first four cases are trivial. The fifth case is where we do the interesting work; the sixth case is similar.

i. $A \equiv B \vee C$, with $\delta(B), \delta(C) \leq n$. We begin by computing $[\underline{t}/x](A^\dagger)$ and $([t/x]A)^\dagger$. First

$$\begin{aligned} [\underline{t}/x](A^\dagger) &\equiv [\underline{t}/x]((B \vee C)^\dagger) \\ &\equiv [\underline{t}/x](B^\dagger \vee C^\dagger) \\ &\equiv [\underline{t}/x](B^\dagger) \vee [\underline{t}/x](C^\dagger), \end{aligned}$$

and then

$$\begin{aligned} ([t/x]A)^\dagger &\equiv ([t/x](B \vee C))^\dagger \\ &\equiv ([t/x]B \vee [t/x]C)^\dagger \\ &\equiv ([t/x]B)^\dagger \vee ([t/x]C)^\dagger. \end{aligned}$$

By the inductive hypothesis we have

$$T' \vdash [\underline{t}/x](B^\dagger) \wedge (t\!\downarrow)^\dagger \to ([t/x]B)^\dagger$$

and

$$T' \vdash [\underline{t}/x](C^\dagger) \wedge (t\!\downarrow)^\dagger \to ([t/x]C)^\dagger$$

so if we define

$$\begin{aligned} D &:= [\underline{t}/x](B^\dagger), \\ E &:= [\underline{t}/x](C^\dagger), \\ F &:= ([t/x]B)^\dagger, \end{aligned}$$

$$G := ([t/x]C)^\dagger,$$
$$H := (t{\downarrow})^\dagger,$$

then we have

$$T' \vdash D \wedge H \to F$$

and

$$T' \vdash E \wedge H \to G$$

and want

$$T' \vdash (D \vee E) \wedge H \to (F \vee G),$$

which is easy to derive.

ii. $A \equiv B \wedge C$. Similar, and easier.

iii. $A \equiv \exists z B$, with $\delta(B) = n$. We begin by computing $[\underline{t}/x](A^\dagger)$ and $([t/x]A)^\dagger$. First

$$\begin{aligned} [\underline{t}/x](A^\dagger) &\equiv [\underline{t}/x]((\exists z B)^\dagger) \\ &\equiv [\underline{t}/x](\exists z B^\dagger) \\ &\equiv \exists z [\underline{t}/x](B^\dagger), \end{aligned}$$

and then

$$\begin{aligned} ([t/x]A)^\dagger &\equiv ([t/x]\exists z B)^\dagger \\ &\equiv (\exists z [t/x]B)^\dagger \\ &\equiv \exists z ([t/x]B)^\dagger. \end{aligned}$$

By the induction hypothesis we have

$$T' \vdash [\underline{t}/x](B^\dagger) \wedge (t{\downarrow})^\dagger \to ([t/x]B)^\dagger$$

so it is an easy proof that $T' \vdash \exists z [\underline{t}/x](B^\dagger) \wedge (t{\downarrow})^\dagger \to \exists z ([t/x]B)^\dagger$.

iv. $A \equiv \forall z B$. Similar.

v. $A \equiv P(\vec{s})$, with $\delta(\vec{s}) = n + 1$. We begin by computing $[\underline{t}/x](A^\dagger)$ and $([t/x]A)^\dagger$. First

$$\begin{aligned} [\underline{t}/x](A^\dagger) &\equiv [\underline{t}/x](P(\vec{s}))^\dagger) \\ &\equiv [\underline{t}/x](P(\underline{\vec{s}} \wedge (\vec{s}{\downarrow})^\dagger) \\ &\equiv [\underline{t}/x]P(\underline{\vec{s}}) \wedge [\underline{t}/x](\vec{s}{\downarrow})^\dagger, \end{aligned}$$

and then

$$\begin{aligned}([t/x]A)^\dagger &\equiv ([t/x]P(\vec{s}))^\dagger \\ &\equiv (P([t/x]\vec{s}))^\dagger \\ &\equiv P(\underline{[t/x]\vec{s}}) \wedge ([t/x]\vec{s}\downarrow)^\dagger.\end{aligned}$$

So it suffices to show two things:

(i) $T' \vdash [\underline{t}/x]P(\underline{\vec{s}}) \wedge [\underline{t}/x](\vec{s}\downarrow)^\dagger \wedge (t\downarrow)^\dagger \to ([t/x]\vec{s}\downarrow)^\dagger$, and
(ii) $T' \vdash [\underline{t}/x]P(\underline{\vec{s}}) \wedge [\underline{t}/x](\vec{s}\downarrow)^\dagger \wedge (t\downarrow)^\dagger \to P(\underline{[t/x]\vec{s}})$.

To that end we assume $\vec{s} = s_0, \ldots, s_n$ is as follows:

$$s_0 \equiv x,$$
$$s_1 \equiv (\iota y)B_1(x,y), \ldots, s_k \equiv (\iota y)B_k(x,y),$$
$$s_{k+1} \equiv v_1, \ldots, s_n \equiv v_{n-k}$$

the v_i variables distinct from x. Now we turn to our two subgoals:

(i) We want to show (in T') that $([t/x]\vec{s}\downarrow)^\dagger$. This is equivalent to

$$[t/x]\left[(x\downarrow)^\dagger \wedge \bigwedge_{1\le i\le k} \exists! y B_i(x,y)^\dagger \wedge \top \wedge \cdots \wedge \top\right]$$
$$\Leftrightarrow (t\downarrow)^\dagger \wedge \bigwedge_{1\le i\le k} \exists! y B_i(t,y)^\dagger.$$

So since we have $(t\downarrow)^\dagger$ as a hypothesis it suffices to show $\bigwedge_{1\le i\le k} \exists! y B_i(t,y)^\dagger$. But we also have the hypothesis that $[\underline{t}/x](\vec{s}\downarrow)^\dagger$; in particular, that

$$[\underline{t}/x]((\iota y)B_i(x,y)\downarrow)^\dagger$$

for each $1 \le i \le k$. This is equivalent to

$$[\underline{t}/x](\exists! y B_i(x,y)^\dagger) \equiv \exists! y [\underline{t}/x](B_i(x,y)^\dagger).$$

So we have $\bigwedge_{1\le i\le k} \exists! y [\underline{t}/x](B_i(x,y)^\dagger)$. Then since

$$\delta(A) = \delta(\vec{s}) \ge \delta((\iota y)B_i(x,y)) > \delta(B_i(x,y)),$$

the inductive hypothesis gives

$$T' \vdash [\underline{t}/x](B_i(x,y)^\dagger) \wedge (t\downarrow)^\dagger \to B_i(t,y)^\dagger$$

for each i, and we are done.

(ii) Finally, we want to show (in T') that $P(\underline{[t/x]\vec{s}})$. Since we have $[\underline{t}/x]P(\vec{s})$, and T' has a substitution rule, it suffices to show that

$$[\underline{t}/x]\underline{s_i} = \underline{[t/x]s_i}$$

for each i.

We get the easy cases out of the way first. For $i = 0$,

$$[\underline{t}/x]\underline{s_0} = [\underline{t}/x]\underline{x} = [\underline{t}/x]x = \underline{t}, \text{and}$$

$$\underline{[t/x]s_0} = \underline{[t/x]x} = \underline{t}.$$

For $k + 1 \leq i \leq n$,

$$[\underline{t}/x]\underline{s_i} = [\underline{t}/x]\underline{v_{i-k}} = [\underline{t}/x]v_{i-k} = v_{i-k}, \text{and}$$

$$\underline{[t/x]s_i} = \underline{[t/x]v_{i-k}} = \underline{v_{i-k}} = v_{i-k}.$$

Finally we consider $1 \leq i \leq k$. In this case we have

$$[\underline{t}/x]\underline{s_i} = [\underline{t}/x]\underline{(\iota y)B_i(x,y)} = [\underline{t}/x]((\iota y)B_i(x,y)^\dagger) = (\iota y)[\underline{t}/x](B_i(x,y)^\dagger),$$

and

$$\underline{[t/x]s_i} = \underline{[t/x](\iota y)B_i(x,y)} = \underline{(\iota y)B_i(t,y)} = (\iota y)B_i(t,y)^\dagger,$$

so we want

$$T' \vdash (\iota y)[\underline{t}/x](B_i(x,y)^\dagger) = (\iota y)B_i(t,y)^\dagger.$$

Since we have $(t\!\downarrow)^\dagger$ and $[\underline{t}/x](\vec{s}\!\downarrow)^\dagger$, which gives $[\underline{t}/x](\exists! y B_i(x,y)^\dagger)$, and have

$$T' \vdash [\underline{t}/x](\exists! y B_i(x,y)^\dagger) \wedge (t\!\downarrow)^\dagger \rightarrow (\exists! y B_i(t,y))^\dagger$$

(by induction), we get the consequent, $(\exists! y B_i(t,y))^\dagger$. Then, replacing $[\underline{t}/x](\exists! y B_i(x,y)^\dagger)$ with its equivalent, $\exists! y[\underline{t}/x](B_i(x,y)^\dagger)$, we have, in summary,

$$\exists! y[\underline{t}/x](B_i(x,y)^\dagger),$$
$$\exists! y B_i(t,y)^\dagger.$$

Using these, together with the description axiom of T', we see that from

$$T' \vdash [\underline{t}/x](B_i(x,y)^\dagger) \wedge (t\!\downarrow)^\dagger \rightarrow B_i(t,y)^\dagger,$$

which we get by induction, we may conclude

$$T' \vdash (\iota y)[\underline{t}/x](B_i(x,y)^\dagger) = (\iota y)B_i(t,y)^\dagger,$$

as desired.

vi. $A \equiv \neg P(\vec{s})$. Similar.

□

Now we can handle the quantificational axioms themselves.

Lemma 11 *For all formulas A and terms t of T^* such that x is free in A and t is free for x in A,*

$$T' \vdash (\forall x A(x) \wedge (t\downarrow) \rightarrow A(t))^\dagger.$$

This follows easily from the last lemma.

Lemma 12 *For all formulas A and terms t of T^* such that x is free in A and t is free for x in A,*

$$T' \vdash (A(t) \wedge (t\downarrow) \rightarrow \exists x A(x))^\dagger.$$

Proof is similar to that of Lemma 11.
And so at last we have:

Theorem 13 *DZFC is a conservative extension of* ZFC′.

3.4.4 Third extension

The third and final conservative extension theorem is very easy. For arbitrary σ (possibly empty), a function symbol $f \notin \sigma$, and a relation symbol $Q \notin \sigma$, we want to prove that

i. DZFC($\sigma \cup \{f\}$) is a conservative extension of DZFC(σ), and

ii. DZFC($\sigma \cup \{Q\}$) is a conservative extension of DZFC(σ).

Theorem 14 DZFC($\sigma \cup \{f\}$) *is a conservative extension of* DZFC(σ)

Proof. We define a translation * from DZFC($\sigma \cup \{f\}$) to DZFC(σ) as follows:

$$\begin{array}{l} ^* \text{ commutes with } \wedge, \vee, \neg, \rightarrow, \forall, \exists; \\ P(\vec{t})^* := P(\underline{\vec{t}}); \\ \underline{x} := x; \\ \underline{(\iota z)B(z)} := (\iota z)B(z)^*; \\ \underline{g(\vec{t})} := g(\underline{\vec{t}}) \text{for} g \neq f; \\ \underline{f(\vec{t})} := (\iota y)A_f(\underline{\vec{t}}, y). \end{array}$$

Note that in the last line we have $A_f(\vec{\underline{t}}, y)$ itself, not $A_f(\vec{\underline{t}}, y)^*$, since A_f is already f-free.

Now it remains to check that every axiom of DZFC($\sigma \cup \{f\}$) translates to a provable formula of DZFC(σ). This time there are no challenges. □

Theorem 15 DZFC($\sigma \cup \{R\}$) *is a conservative extension of* DZFC(σ)

Proof. Similar to the last one, only easier. This time the translation * is:

$$
\begin{array}{l}
^*\text{commuteswith} \wedge, \vee, \neg, \rightarrow, \forall, \exists; \\
P(\vec{t})^* := P(\vec{\underline{t}}) \text{for} P \neq Q; \\
Q(\vec{t})^* := \varphi_Q(\vec{\underline{t}}); \\
\underline{x} := x; \\
\underline{(\iota z)B(z)} := (\iota z)B(z)^*; \\
\underline{g(\vec{t})} := g(\vec{\underline{t}}).
\end{array}
$$

□

From Theorems 14 and 15 we get:

Theorem 16 *For any set σ of defined functions and relations,* DZFC(σ) *is a conservative extension of* DZFC.

Finally, by transitivity we have:

Theorem 17 *For any set σ of defined functions and relations,* DZFC(σ) *is a conservative extension of* ZFC.

3.5 Sample input-output pairs

We have just finished defending DZFC, and showing what a good system it is to work in. Now we will show what a bad language it has!

In this section we revisit some of the examples we saw in Section 2.2, this time viewing the input LPT and LaTeX LPT along with the output of `lpt2dzfc`, i.e. the translation into the language of DZFC. Comparing the DZFC output side by side with the LPT input, the reader can see the advantages of working in LPT.

Example 1. First we revisit our example of the usage of the description operator to bind an ordered pair. Whereas in LPT we are able to refer to "the unique ordered pair $\langle Y, T' \rangle$ such that...," this translates to a much clumsier expression in DZFC.

```
DEFINITION MunkTop.29.4: 2-ary function Oneptcompactification.
If TOPSP[X,T] then Oneptcompactification(X,T) \simeq
(!<Y,T'>)(
  COMPACTIFICATION[Y,T',X,T] \wedge Y \less X \approx_{C} 1_{N}
).
```

DEFINITION MunkTop.29.4: 2-ary function `Oneptcompactification`. If $\texttt{TOPSP}[X, T]$ then $\texttt{Oneptcompactification}(X, T) \simeq (!\langle Y, T'\rangle)(\texttt{COMPACTIFICATION}[Y, T', X, T] \wedge Y \backslash X \approx_{\texttt{C}} 1_{\texttt{N}})$.

$\texttt{Oneptcompactification}(X, T) \simeq (\iota y_0)(\texttt{TOPSP}[X, T] \wedge y_0 \simeq (\iota x_0)(\exists Y, T')(x_0 = \varpi_{\texttt{0}}(Y, T') \wedge (\texttt{COMPACTIFICATION}[Y, T', X, T] \wedge \approx_{\texttt{C}}[\backslash(Y, X), 1_{\texttt{N}}])))$

Example 2. Next observe what happens in DZFC, where we cannot match the brevity of expression used in our definition of the `FCN` predicate in LPT.

```
DEFINITION FS.2.58: 1-ary relation FCN. FCN[f] \iff
f = {<x,y> : f(x) = y}.
```

DEFINITION FS.2.58: 1-ary relation `FCN`. $\texttt{FCN}[f] \leftrightarrow f = \{\langle x, y\rangle : f(x) = y\}$.

$\texttt{FCN}[f] \leftrightarrow f = (\iota z_0)(\forall y_0)(y_0 \in z_0 \leftrightarrow (\exists x, y)(y_0 = \varpi_{\texttt{0}}(x, y) \wedge ((\iota x_0)(\varpi_{\texttt{0}}(x, x_0) \in f) = y)))$

Example 3. Here we see the result of giving up our lambda operator:

```
DEFINITION MunkTop.19.2.5: 2-ary function Cartespow. Cartespow(A,B)
\simeq Cartesprod((\lambda b \in B)(A),B).
```

DEFINITION MunkTop.19.2.5: 2-ary function `Cartespow`. $\texttt{Cartespow}(A, B) \simeq \texttt{Cartesprod}((\lambda b \in B)(A), B)$.

$\texttt{Cartespow}(A, B) \simeq \texttt{Cartesprod}((\iota z_0)(\forall y_0)(y_0 \in z_0 \leftrightarrow (\exists b, x_0)(y_0 = \varpi_{\texttt{0}}(b, x_0) \wedge x_0 = (A) \wedge b \in B)), B)$

Chapter 4

Second application: Structural observations

In this chapter we review various structural data culled from the database of definitions in Appendices F and G. In all cases, our data must be understood to reflect not a necessary logical structure of mathematical concepts, but a historically factual structure that they have been given, through actual definitions that have been used.

At the same time, different significance must be associated with our different measures: one may be language-dependent, while another may be independent of the language in which the definitions are given. In particular, in the first section below we consider a pair of structural measures: the depth and size of each definition's DAG, or directed acyclic graph of conceptual dependencies. These are independent of the choice of language. On the other hand, in the subsequent section we investigate quantifier depth, which is dependent upon the choice of language.

Which, if any, of our measures are good measures of the complexity of concepts? A good way to answer this would be to scan the definitions in our database, and compare each against the various structural measures we have investigated. Then, consulting our intuition as to which are the "complex" definitions, we could judge which measures gave the best reflection of complexity.

In the sections below we first consider DAGs, then quantifier nesting depth, then raw symbol count, and finally the types (function or relation, plus arity) of definienda occurring in our database.

4.1 Dags

Following Friedman and Flagg in [9], we define the definition DAG (Directed Acyclic Graph) associated to a defined concept α. Intuitively, we begin with the definition *tree* for α, and then identify common subtrees to get the DAG.

For the precise definition of a DAG, let Ω be the set of all definienda in our database. Given $\alpha, \beta \in \Omega$, we say that α *depends directly on* β, and write

$$\alpha \to \beta$$

just in case β appears in the definiens for α.

We say that α *depends on* β, and write

$$\alpha \Rightarrow \beta$$

if and only if there exists a sequence $\chi_1, \ldots, \chi_n \in \Omega$ such that

$$\alpha = \chi_1 \to \chi_2 \to \cdots \to \chi_n = \beta.$$

Then the definition DAG $\mathscr{D}(\alpha)$ for a definiendum α, is the graph having as vertices the set $V(\mathscr{D}(\alpha))$ of all $\beta \in \Omega$ such that $\alpha \Rightarrow \beta$, and such that for all $\beta, \gamma \in V(\mathscr{D}(\alpha))$ there is an edge from β to γ just in case $\beta \to \gamma$.

We will measure two aspects of each definition DAG. For one, we count the number of vertices appearing in the DAG. We call this the *size*. We call our second measurement the *depth* of the DAG, and it is defined to be the greatest n for which there is a sequence $\chi_1, \ldots, \chi_n \in V(\mathscr{D}(\alpha))$ with

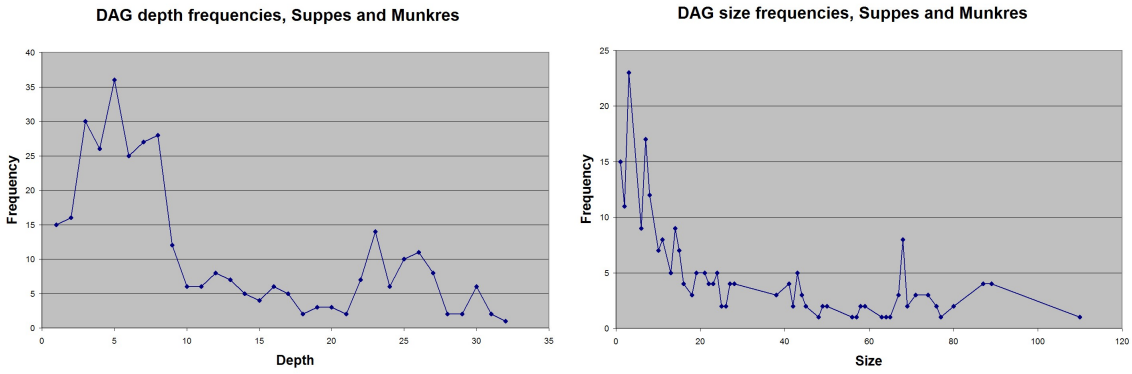
$$\alpha = \chi_1 \to \chi_2 \to \cdots \to \chi_n.$$

We prefer a definition's DAG to its tree of conceptual dependencies first of all because, while the two will have the same depth, the size of the DAG is a more easily interpreted measure than the size of the tree. The former tells us how many different concepts are involved, whereas the latter is in general a larger number, with a less obvious meaning.

Secondly, we prefer to develop tools to work with the DAG since it is generally a smaller and much more manageable graph, which is preferable for applications such as the one discussed in Chapter 5.

We review now some findings on the depths and sizes of the DAGs for the definitions taken from Suppes (Appendix F) and from Munkres (Appendix G). The full list of depth and size data can be found in Appendix C.

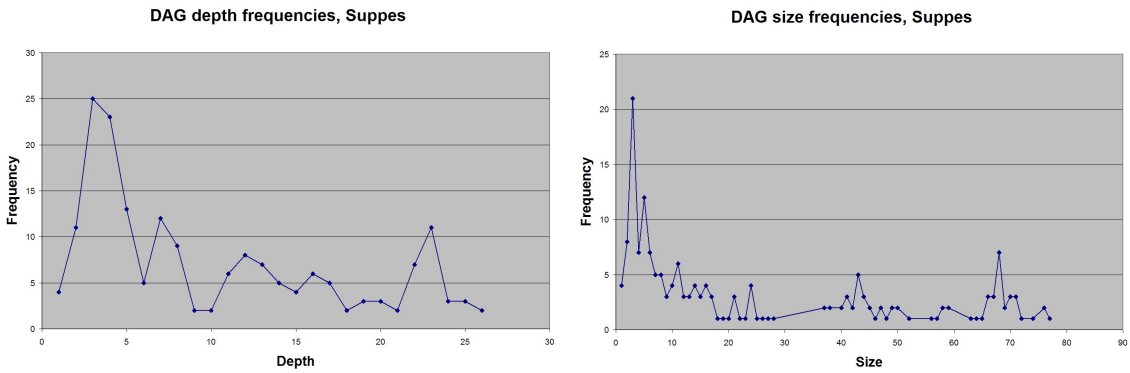
To begin with, consider the histograms for depths and sizes taken across all definitions, from both Suppes and Munkres.

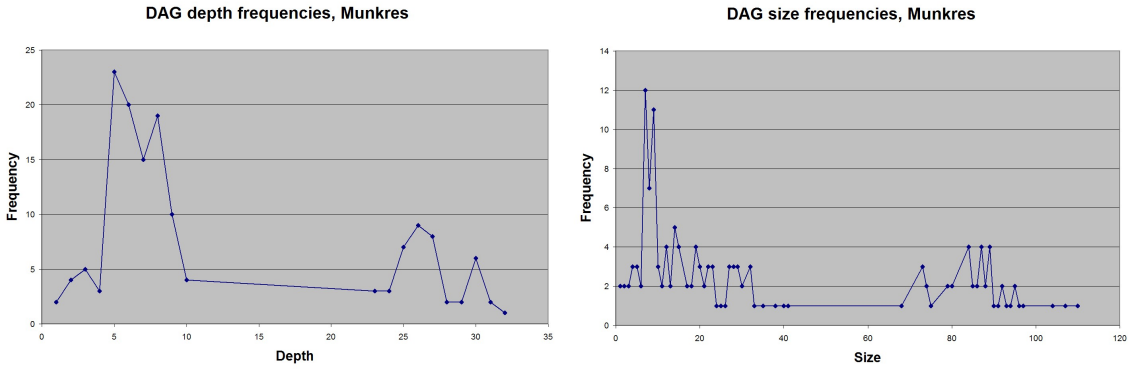


It is interesting to note in the depth histogram that 5 is the most common depth, and that depths 3 through 8 are together very common. The graph has a two-humped shape. The second hump (depths 22 through 27) corresponds to the Suppes definitions that develop the real numbers, as well as the Munkres definitions that mention the real numbers.

In the size histogram we see that 3 and 7 are the two most common sizes. The graph falls off steadily, again with the definitions related to the real numbers responsible for the high numbers.

We may also consider the depth and size histograms for the Suppes and Munkres definitions separately, as below:





From these data we can extract the maxima, and the means:

		Max	Mean
All	Depth	32	10.77
	Size	110	29.56
Suppes	Depth	26	10.09
	Size	77	25.91
Munkres	Depth	32	12.25
	Size	110	36.01

Note that DAG measurements such as depth and size are reflections not of the language in which we have chosen to write our definitions, but only of the definitional structure we have chosen.

Something that we see reflected in the depth numbers for the Munkres definitions (in Appendix C), is that any definition involving mention of the real numbers is inordinately deep. Consider for example the definition of the standard topology on the reals, in Chapter 13 (Stdrealtop:0), which has depth 24.

This depth reflects all the work that goes into formally defining the reals. To a student, on the other hand, this depth has nothing to do with the difficulty of the concept; to the student, the real numbers bring in no conceptual difficulty whatsoever. Quite the contrary, their involvement is a conceptual aid, in that it allows the production of a concrete example.

4.2 Quantifier depths

In this section we examine the quantifier depths occurring in the definitions in our database. Quantifier depth can be regarded in two different ways. Consider the

following example:

DEFINITION MunkTop.21.3: 7-ary relation UNIFCONV. If $\texttt{TOPSP}[X,T] \wedge \texttt{METRICSPACE}[Y,T',d] \wedge \texttt{FCN}[F] \wedge \texttt{Dom}(F) = \mathbb{N} \wedge (\forall n \in \mathbb{N})(\texttt{FCN}[F(n),X,Y])$ then $\texttt{UNIFCONV}[F,f,X,T,Y,T',d] \leftrightarrow (\forall \varepsilon \in \mathbb{R})(\varepsilon >_{\texttt{R}} 0_{\texttt{R}} \rightarrow (\exists N \in \mathbb{N})((\forall n >_{\texttt{N}} N)(\forall x \in X)(d(F(n)(x), f(x)) <_{\texttt{R}} \varepsilon)))$.

The $<_{\texttt{R}}$ relation occurs inside of a universal quantifier, followed by an existential quantifier, and then two universal quantifiers. Counting only quantifier alternations, we would say then that the $<_{\texttt{R}}$ relation occurs at depth 3. If on the other hand we count any and all quantifiers, whether or not they alternate, then the $<_{\texttt{R}}$ relation occurs at depth 4.

Alternating depth is typically of greater interest as logical structure is concerned. Consider for example the arithmetic hierarchy. I believe however that non-alternating depth is also of interest, at least psychologically. The two subsequent universal quantifiers in the example above introduce two different *kinds* of things that the reader must keep track of: a natural number greater than N, and an element of the set X. This is more complex than what would be introduced by a single quantifier.

We will measure both alternating, and non-alternating, quantifier depth for each definition. In addition, we will regard each definition at three different levels of *expansion*.

To expand a definition is to replace each defined function or relation appearing in the definiens by its own definiens. This process may be repeated recursively until it halts. In this case we will say that the given definition was expanded fully.

In general, let Ω be the set of all definienda in our database. Given $\alpha \in \Omega$, and a set $B = \{\beta_1, \ldots, \beta_n\} \subseteq \Omega$, we define *the expansion of α relative to B*, which we denote $\mathscr{E}(\alpha, B)$, to be what we get by replacing all definienda in α, except those in B, by their definiens; and repeating this process until it halts.

Altogether, in this section we consider for each definition α in our database, eight data points: Both alternating, and non-alternating, quantifier depth, for α viewed as given in LPT; as translated into DZFC; and then as the translation into DZFC is both fully expanded, and expanded relative to the set of the following foundational definitions:

DEFINITION builtIn1: Infix function $\cup$. $x \cup y \simeq \{z : z \in x \vee z \in y\}$. Precedence 40.

DEFINITION builtIn2: Infix function $\cap$. $x \cap y \simeq \{z : z \in x \wedge z \in y\}$. Precedence 30.

DEFINITION builtIn3: Infix function $\setminus$. $x \setminus y \simeq \{z : z \in x \wedge \neg(z \in y)\}$. Precedence 50.

DEFINITION builtIn4: 2-ary function ϖ_0. $\varpi_0(a, b) \simeq \{\{a\}, \{a, b\}\}$.

DEFINITION builtIn5: Infix relation $\subseteq$. $X \subseteq Y \leftrightarrow (\forall x)(x \in X \rightarrow x \in Y)$.

DEFINITION builtIn6: 0-ary function $\varnothing$. $\varnothing \simeq (!S)(\forall x)(\neg(x \in S))$.

DEFINITION builtIn7: 1-ary function $\cup$. $\cup(X) \simeq \{u : (\exists x \in X)(u \in x)\}$.

DEFINITION builtIn8: 1-ary function $\cap$. $\cap(X) \simeq \{u : (\forall x \in X)(u \in x)\}$.

DEFINITION builtIn9: 1-ary function $\wp$. $\wp(X) \simeq \{U : U \subseteq X\}$.

DEFINITION builtIn10: Infix relation $\supseteq$. $X \supseteq Y \leftrightarrow (\forall y)(y \in Y \rightarrow y \in X)$.

The thought is that these low, foundational definitions, are apt to make a large contribution to the depth of an expanded definition, if they too are expanded in the process. It will be interesting to see what results from leaving them unexpanded, especially since mathematicians always read sentences of set theory without ever asking for a reminder of the definition of these concepts.

The tables listing all eight data points for each definition can be found in Appendix D. In the section below we consider a few observations on this data. In the subsequent section, I discuss the algorithm I used to compute quantifier depths.

4.2.1 Quantifier data

Consider first the maximum and mean values.

	Max	Mean
LPT	4	0.66
unexpanded DZFC	5	1.31
fully expanded DZFC	1235	78.68
partially expanded DZFC	552	38.54
LPT alternating	3	0.63
unexpanded DZFC alternating	5	1.18
fully expanded DZFC alternating	422	36.19
partially expanded DZFC alternating	239	22.16

The conjecture has been made that among actually occurring definitions, in mathematics texts, the maximum alternating quantifier depth is three. Considering that

among our data, the LPT input comes closest to what actually occurs in textbooks, the maximum alternating depth of 3 for LPT tends to confirm this conjecture.

Note that the maximum depth after translating into DZFC goes up to 5. This reflects what we saw in Section 3.5. In each of the examples in that section, we saw that a definition that used no quantifiers in LPT turned out to require them after translation into DZFC.

The maximum depth of 1235 for a fully expanded definition confirms the necessity of using definitions to package information into manageable chunks. Meanwhile, the contrast between the total expansion maximum, and the partial expansion maximum, demonstrates that the lowest, most foundational definitions, lend quite a bit of this complexity.

It is also interesting to observe the difference between the fully expanded non-alternating depth, and the fully expanded alternating depth. That the latter is so much smaller than the former indicates that somewhere in the full expansions there are long strings of universal, or existential, quantifiers, without alternation.

Consider the mean depth for LPT alternating (again, what comes closest to what we ordinarily think of as quantifier depth in textbooks). This shows that, while the maximum is three, the most common depths are 0 and 1. In fact, the number of occurrences are as in the following table.

	Occurrences	
Depth	LPT	LPT alt.
0	178	178
1	118	120
2	30	35
3	14	8
4	1	0

4.2.2 Calculating quantifier depths for expanded definitions

The most obvious algorithm with which to calculate the maximum quantifier depth occurring in an expanded definition begins by fully expanding a given definition, and then scans the result for maximum quantifier depth. But this is infeasible in practice.

A feasible alternative is to record for each definition d of a k-ary function or relation α a vector $\langle a_0, a_1, \ldots, a_k \rangle$ of integers indicating that if $x_0, x_1, \ldots, x_{k-1}$ are the k free variables appearing in d, then a_i is the maximum quantifier depth at which x_i occurs, for $0 \leq i < k$, and that a_k is the maximum quantifier depth occurring anywhere in d. We call this the *depth vector* $A(d)$ for d.

Then we may compute the quantifier depth of d as a function of the depths of the k arguments. We define the function

$$D(d)(x_0, \ldots, x_{k-1}) := \max(a_0 + x_0, \ldots, a_{k-1} + x_{k-1}, a_k).$$

Then if the quantifier depths of the k arguments are $b_0, \ldots, b_{k-1}$, then the depth of d when we plug in these arguments is $D(d)(b_0, \ldots, b_{k-1})$.

Now suppose $d_0, d_1, \ldots, d_n$ are the definitions for which we wish to compute the quantifier depth. Assume that d_0 has no conceptual dependencies (its DAG has depth 1), and that for $i > 0$ the set of concepts on which d_i depends is contained in $\{d_0, \ldots, d_{i-1}\}$. Then we may proceed by first computing the depth vector $A(d_0)$ for d_0 and storing it. Then we return $D(d_0)(0, \ldots, 0)$, the maximum quantifier depth occurring in d_0.

We then proceed similarly for the remaining d_i, consulting our stored depth vectors whenever we encounter a defined function or relation. With this algorithm we can feasibly compute the quantifier depths for all of the definitions in our database.

A slightly modified algorithm is needed in order to compute the alternating quantifier depths. The basic process is the same, only the depth vectors need to contain more information. This time, if the defined function or relation is k-ary, then, instead of a vector of length $k + 1$, we will store a vector of length $4k + 2$, the entries being as follows:

i. For each of the k free variables, x_i, we record the maximum number of alternating nested quantifiers under which x_i occurs, such that:

   (i) the first quantifier is $\exists$ and the last is also $\exists$,

   (ii) the first is $\exists$ and the last is $\forall$,

   (iii) the first is $\forall$ and the last is $\exists$,

   (iv) the first is $\forall$ and the last is also $\forall$.

ii. We record the maximum depth of quantifiers occurring anywhere in the definition and starting with $\exists$

iii. We record the maximum depth of quantifiers occurring anywhere in the definition and starting with $\forall$.

We must then follow a somewhat more complex process in order to use these depth vectors, but otherwise the algorithm is essentially the same as in the non-alternating case.

4.3 Symbol counts

I have also measured the raw symbol count for each definition, regarded (1) as its translation into DZFC; (2) as fully expanded; and (3) as partially expanded, relative to the same set of foundational definitions given in Section 4.2. To obtain this data I used a similar algorithm to the one described in Section 4.2.2.

All the data can be found in Appendix E. Again, the contrast between full and partial expansion can be seen, as in the last section. But both numbers go exponentially, and rapidly become meaningless, as they become so large.

What is of most interest from the raw symbol counts is just the number of symbols in the unexpanded definitions. This at least gives us some sense of the size of definitions occurring in texts. The longest definitions in my database were 526 symbols, occurring in Suppes Chapter 6; and 714 symbols, occurring in Munkres Section 33.

4.4 Types

Following is the list of all types that occurred in the definitions compiled in Appendices F and G, a type being a pair in which the first component is either *function* or *relation*; and in which the second component is an arity: either *infix*, or a non-negative integer.

Relations								
arity	infix	1	2	3	4	5	6	7
occurrences	35	32	57	40	9	7	1	1

Functions						
arity	infix	0	1	2	3	4
occurrences	23	42	36	20	16	9

Chapter 5

Third application: Knowledge graphs

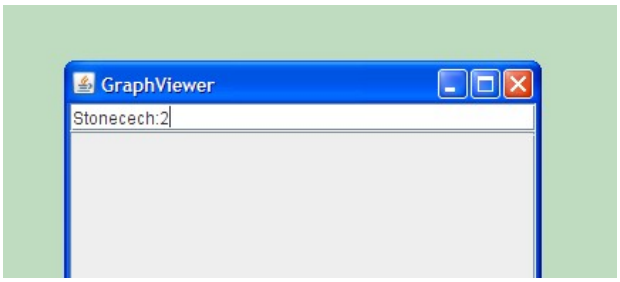
To a student exploring a new field of mathematics, the defined concepts of the field represent a first hurdle that must be cleared. The language cannot even be understood until the concepts are. For a dedicated student beginning to study a well-defined subject, a traditional approach using a textbook may be best. But it is difficult to look ahead in a textbook, to get a quick view of what is to come. A student wishing to look ahead, or even an expert from another field dabbling in a foreign subject, will benefit from software that allows the quick diagrammatic representation of conceptual dependencies.

Suppose for example that someone knowing nothing about modularity theory wants to understand the concepts involved in the proof of Fermat's Last Theorem. They will locate a book on elliptic curves, and attempt to find out what it means for an *elliptic curve* to be *modular*. They will find the definitions of these concepts in the book, and discover that these depend on prior concepts, also unknown. And so a backwards chase through the book is initiated.

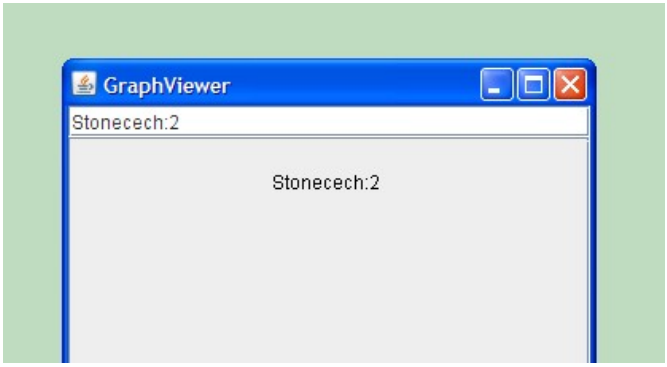
My program, `kmap` (for "knowledge map"), addresses the needs of such a student. In the first section below, I go over its usage. In the following section I discuss future improvements.

5.1 kmap

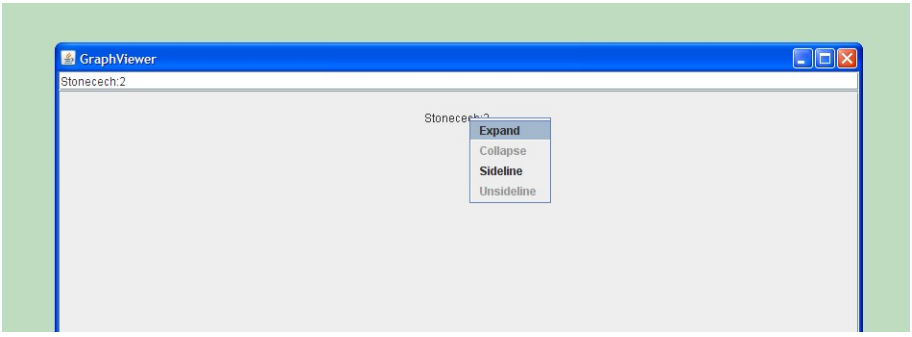
The user of `kmap` begins by typing in the name and arity of the definition $\mathscr{D}$ to be explored, and pressing `Enter`.



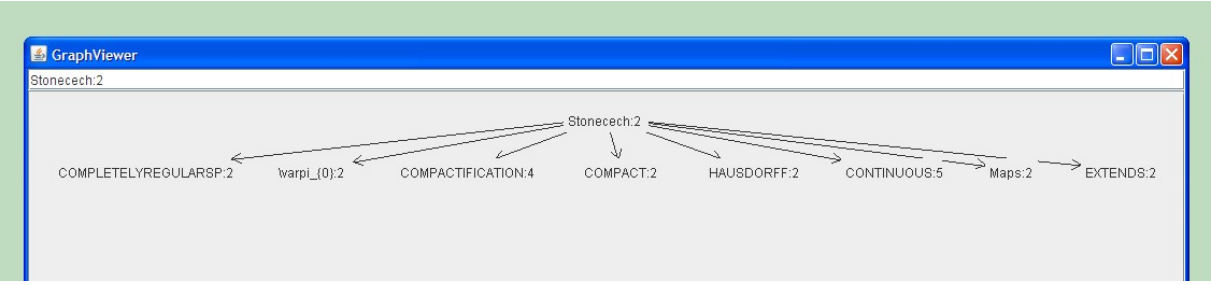
This causes the root node of the definition DAG for $\mathscr{D}$ to be displayed in the GraphViewer window.



The user then has the option to click on nodes in the GraphViewer window, and choose either to "expand" or "collapse" them.

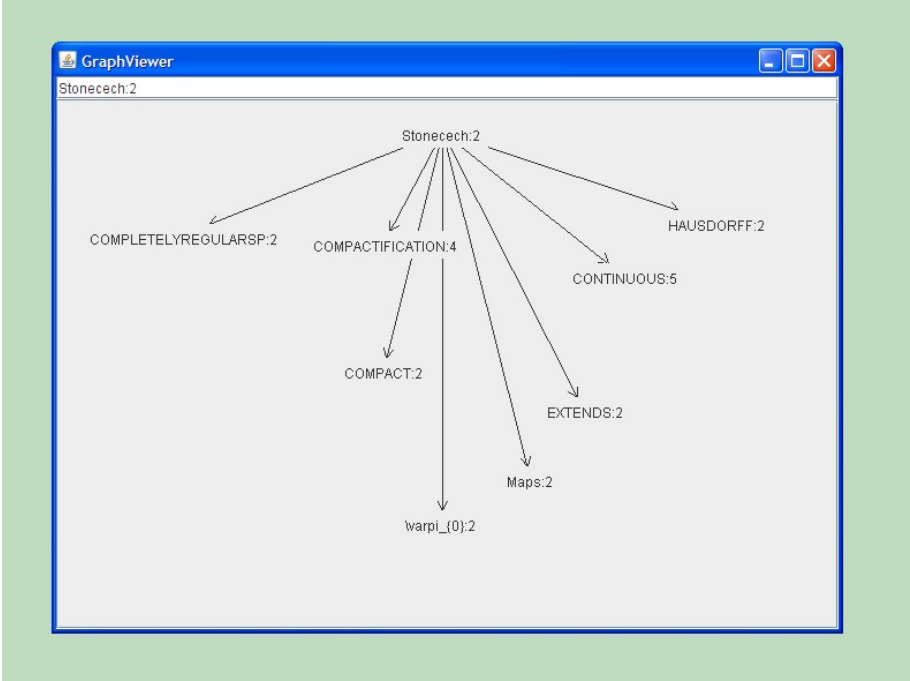


Expanding a node causes all of the concepts on which it directly depends (i.e. all those concepts mentioned explicitly in its definiens) to be displayed beneath it, with directed edges drawn in to show the dependencies.



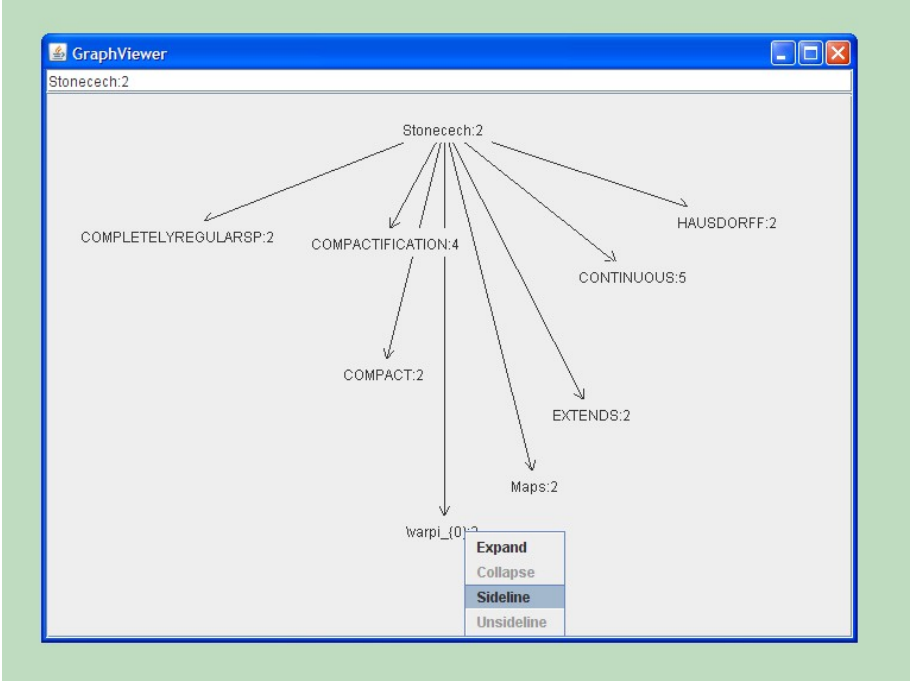
Collapsing a node deletes all nodes that are depended upon only by it.

By clicking and dragging with the mouse, the user can rearrange the nodes of the graph.

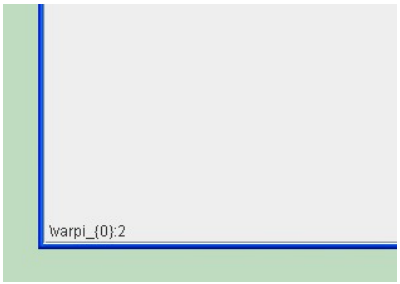


Since many of the concepts arising in the expansion process will be well-known, and therefore boring, to a student, my program also offers the option to "sideline" a concept.

For example, in the figure above, the lowest node displays our name for the ordered pair function. This can be sidelined, in order to remove clutter from the diagram. The user right-clicks on the node, and selects "Sideline".



This causes the node to be relocated to the left-hand margin of the GraphViewer window, and all edges connecting the node to the rest of the graph to be (temporarily) deleted.



The user is free to "unsideline" any sidelined concepts at any time, returning them to their prior location within the graph.

5.2 Future directions

Names. The sorts of names we have used for the definitions in Appendix G are somewhat difficult to read. After some time away from them, a user could forget what they abbreviate.

A better alternative is to work within various *namespaces*, so that what is extraneous or hard to read can be removed from the names. For example, if it was known that we were working in the *Topology* context, then we could simply define *convergent*, rather `TOPCONV`.

Generally, within a program like `kmap`, it would be an improvement to see more readable tags on the vertices. Instead of the sort of highly abbreviated names we have used here, it would be good to see a more complete description of the concept being referenced.

Other vertex types. As we explore the DAGs for defined concepts, while keeping in mind the goal of developing software that will allow a student to explore a new area of mathematics, it becomes apparent what is missing. In addition to *concepts*, the Knowledge Map environment should also allow us to explore the network of *theorems* and *proofs* that make up a theory. (To incorporate proofs we would have to choose a language in which to represent them, which LPT does not provide.)

Ultimately we should have Knowledge Maps that feature three kinds of vertices, corresponding to theorems, proofs, and concepts. The edges would be as follows:

1. **Theorem → Proof.** This indicates that the proof is a proof of this theorem.
2. **Theorem → Concept.** This indicates that the concept is mentioned in the statement of the theorem.
3. **Proof → Theorem.** The theorem is cited in the course of the proof.
4. **Proof → Concept.** The concept is used in the course of the proof.
5. **Concept → Concept.** This is the sort of edge we have already considered in our definition DAGs, indicating that the first concept is defined in terms of the second one.

Edges of types 2 and 5 in the list above could be computed by a parsing program, as has been done in my program. Edges of type 1 would perhaps represent data input by hand, or else could follow from the name of the proof, if appropriate naming conventions were set up. Edges of types 3 and 4 would, I believe, have to be input by hand, and in fact this would require mathematical expertise.

Appendix A

Characters and Strings in LPT

A.1 Characters

LPT features a large character set, which we divide into several subsets as follows.

1. English letters: Capital and lowercase, from A to z. These characters may be entered inside of any of the following LaTeX font commands:

   `\mathbf{}` `\mathcal{}` `\mathfrak{}` `\mathscr{}` `\mathbb{}`

2. Greek letters: Capital and lowercase, from Γ to ω (except λ, π, and ω, which are reserved for other purposes), plus the usual LaTeX variations: ε, ϑ, ϖ, ϱ, ς, φ.

α	`\alpha`	β	`\beta`	γ	`\gamma`	δ	`\delta`
ϵ	`\epsilon`	ε	`\varepsilon`	ζ	`\zeta`	η	`\eta`
θ	`\theta`	ϑ	`\vartheta`	ι	`\iota`	κ	`\kappa`
μ	`\mu`	ν	`\nu`	ξ	`\xi`		
ϖ	`\varpi`	ρ	`\rho`	ϱ	`\varrho`	σ	`\sigma`
ς	`\varsigma`	τ	`\tau`	υ	`\upsilon`	ϕ	`\phi`
φ	`\varphi`	χ	`\chi`	ψ	`\psi`	Γ	`\Gamma`
Δ	`\Delta`	Θ	`\Theta`	Λ	`\Lambda`	Ξ	`\Xi`
Π	`\Pi`	Σ	`\Sigma`	Υ	`\Upsilon`	Φ	`\Phi`
Ψ	`\Psi`	Ω	`\Omega`				

3. Digits: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.

4. Function characters: Here we include many of the popular function symbols in AMS LaTeX.

π	\pi	ω	\omega	$+$	+	\	\less
/	/	|	|	$\cup$	\cup	$\cap$	\cap
$\sqcup$	\sqcup	$\sqcap$	\sqcap	$\oplus$	\oplus	$\otimes$	\otimes
$\times$	\times	$\div$	\div	$\circ$	\circ	$\cdot$	\cdot
$\varnothing$	\varnothing	∞	\infty	$\wp$	\wp	$\sum$	\sum
$\prod$	\prod	$\coprod$	\coprod	$\bigcup$	\bigcup	$\bigcap$	\bigcap
$\bigsqcup$	\bigsqcup	$\bigvee$	\bigvee	$\bigwedge$	\bigwedge	$\bigotimes$	\bigotimes
$\bigoplus$	\bigoplus	$\aleph$	\aleph	$\beth$	\beth	$\gimel$	\gimel
$\daleth$	\daleth	$\Re$	\Re	$\Im$	\Im	∂	\partial
∇	\nabla						

5. Relation characters: Likewise, we include many of the popular relation symbols in AMS LaTeX.

$\leq$	\leq	$\geq$	\geq	$\prec$	\prec	$\succ$	\succ
$\preceq$	\preceq	$\succeq$	\succeq	$\ll$	\ll	$\gg$	\gg
$\subset$	\subset	$\supset$	\supset	$\subseteq$	\subseteq	$\supseteq$	\supseteq
$\sqsubseteq$	\sqsubseteq	$\sqsupseteq$	\sqsupseteq	$\in$	\in	$\vdash$	\vdash
$\dashv$	\dashv	$\equiv$	\equiv	$\models$	\models	$\sim$	\sim
$\perp$	\perp	$\mid$	\mid	$\asymp$	\asymp	$\parallel$	\parallel
$\approx$	\approx	$\smile$	\smile	$\cong$	\cong	$\frown$	\frown
$\propto$	\propto	$\doteq$	\doteq	$\bowtie$	\bowtie	$\lhd$	\lhd
$\rhd$	\rhd	$\unlhd$	\unlhd	$\unrhd$	\unrhd		

6. Decorating characters: The usual LaTeX decorations, plus primes and asterisks.

$\hat{a}$	\hat{a}	$\tilde{a}$	\tilde{a}	$\bar{a}$	\bar{a}	$\vec{a}$	\vec{a}
$\dot{a}$	\dot{a}	a'	a'	a^*	a^*		

7. Reserved characters: The logical connectives and quantifiers, $=$, $\neq$, braces, brackets, etc. The Greek letter λ is reserved for lambda abstraction.

$\wedge$	\wedge	$\vee$	\vee	$\rightarrow$	\implies	$\neg$	\neg
$\forall$	\forall	$\exists$	\exists	$\top$	\top	$\bot$	\bot
$\leftrightarrow$	\iff	$=$		$\neq$	\neq	$\simeq$	\simeq
$\uparrow$	\up	$\downarrow$	\dn	$<$		$>$	
[		]		(		)	
{		}		λ	\lambda	!	

A.2 Strings

Here we describe the formation of the strings to be used in LPT for the naming of defined functions and relations, as well as variables.

Function strings: A function string consists of

i. Two or more English letters, the first of which is capital and the rest of which are lowercase, or

ii. A function symbol, or

iii. A numeral, or

iv. A Greek letter

followed by an optional subscript and/or primes or an asterisk. The subscript is entered with an underscore and braces as in LaTeX, and may contain any string of characters in the language. In cases (iii) and (iv) there must be a subscript or decorating character.

Examples: `Stonecech`, $+$, $\sqcup$, 0_{Nat}.

Relation strings: A relation string consists of

i. Two or more English letters, all of which are capital, or

ii. A relation symbol

followed by an optional subscript and/or primes or an asterisk.

Examples: `GROUP`, $\in$, $<_{Nat}$.

Variable strings: A variable string consists of

i. An English letter in any font, or

ii. A Greek letter,

where the letter is given any of the five LaTeX decorations mentioned in point vi above, as well as zero or more primes, an optional asterisk, and/or an optional subscript consisting of a nonnegative numeral.

Appendix B

Our LPT grammar is not LL

Let $\mathscr{G}$ be the grammar for LPT implemented by our parsing program, as compiled by ACCENT. Using the techniques and notations of [3], we prove that $\mathscr{G}$ is not LL.

In the first section below, we give the proof. In the subsequent section we present the entire grammar for LPT, as passed to our compiler compiler, ACCENT.

B.1 Proof

We begin by introducing abbreviations for the following nonterminals and terminals of $\mathscr{G}$. (See the next section for the entire grammar.)

Nonterminals

abbreviation	nonterminal	abbreviation	nonterminal
S	`def_list`	A	`definition`
B	`defname`	C	`defbody`
D	`numeral`	E	`rstr`
F	`relndef`	G	`relndefunit`
H	`varseq`	I	`formula`
J	`var`	K	`bformula`
L	`termformula`	M	`term`
Q	`bterm`	R	`aterm`

Terminals

abbreviation	terminal
a	`DEFINITION`
b	`ARY`
c	`RELN`
d	`RELNSYM`
e	`IFF`
f	`LETTER`
g	`TOP`
h	`UPARROW`

The nonterminal S is the start symbol of the grammar.

Apart from the terminals listed above, we also have:

$$0 \quad . \quad (\quad) \quad [\quad]$$

Given these abbreviations, the following productions (among many others) belong to the grammar:

$S \to A$	$E \to d$	$J \to f$	$M \to Q$
$A \to aBC$	$F \to G.$	$K \to (I)$	$Q \to R$
$B \to 0$	$G \to E[H]eI$	$K \to g$	$R \to (M)$
$C \to DbcE.F$	$H \to J$	$K \to L$	$R \to D$
$D \to 0$	$I \to K$	$L \to Mh$	

Now let an arbitrary non-negative integer k be given. It is easy to see that with

$$\begin{aligned} w &= a00bcd.d[f]e \\ \alpha &= . \\ \beta &= (I) \\ \gamma &= L \\ x &= \underbrace{((\cdots(}_{k} \end{aligned}$$

we have

$$\begin{aligned} &S \underset{lm}{\overset{*}{\Rightarrow}} xK\alpha \\ &K \to \beta \text{ and } K \to \gamma \\ &x \in \mathrm{FIRST}_k(\beta\alpha) \cap \mathrm{FIRST}_k(\gamma\alpha) \end{aligned}$$

whence it follows, from Theorem 5.2 of [3], that $\mathscr{G}$ is not $LL(k)$. Since k was arbitrary, $\mathscr{G}$ is not LL.

B.2 The grammar

We reproduce the source code sent to ACCENT, from which it produced an Earley algorithm parser for LPT. The reader may consult [14] regarding the syntax for ACCENT. Briefly, the source code begins with a list of the tokens of the language, which stand for terminal symbols, in all capitals at the beginning. (Other, single-character terminal symbols can be used without a token to represent them.) Thereafter we have a paragraph for each nonterminal, in which its productions are given in Backus-Naur form.

```
%prelude {
#include "yystype.h"
#include "stdio.h"

FILE *pFile;
char AST[10000];
char UP[] = "$<";
char DN[] = "$>";
}

%token ALLCAPS, ARY, BOT, DECORATION,
DEFINITION, DIGITS, DOWN, EXISTS, FIXED, FONT,
FORALL, FUNC, FUNCSYM, GENWORD, GREEK,
IF, IFF, INFIX, INFIXFN, INFIXRL, INITCAP,
LAMBDA, LETTER, LETTERS, MATHBB, MISCLETTER,
NEG, NEQ, NZNUM, OTHERWISE, PRECEDENCE,
RELN, RELNSYM, SIMEQ, THEN, THENSYM, TOP,
UPARROW, VEE, WEDGE;

// -------------------- TRUNK -------------------------

def_list :
  { pFile = fopen("lptout","w"); }
  definition { fprintf(pFile, "%s", AST); strcpy(AST,""); }
  ( { strcat(AST,"\n\n"); }
    definition { fprintf(pFile, "%s", AST); strcpy(AST,""); }
  )*
  { fclose(pFile); }
```

```
;

// ------------------------------------------------------

definition :
  %prelude { strcat(AST,DN); strcat(AST,"definition"); }
  DEFINITION defname (defbody|infixdefbody)
  { strcat(AST,UP); }
;

// ------------------------------------------------------

defname :
  //Definition names are made from alphanumeric characters,
  //as well as the period and underscore characters.
  %prelude { strcat(AST,DN); strcat(AST,"defname"); }
  { strcat(AST,"$$"); }
  (  '0' { strcat(AST,"0"); }
   | '_' { strcat(AST,"_"); }
   | NZNUM<s> { strcat(AST,s); }
   | DIGITS<s> { strcat(AST,s); }
   | LETTER<s> { strcat(AST,s); }
   | INITCAP<s> { strcat(AST,s); }
   | ALLCAPS<s> { strcat(AST,s); }
   | LETTERS<s> { strcat(AST,s); }
   | GENWORD<s> { strcat(AST,s); }
   | '.' { strcat(AST,"."); }
  )
  (  '0' { strcat(AST,"0"); }
   | '_' { strcat(AST,"_"); }
   | NZNUM<s> { strcat(AST,s); }
   | DIGITS<s> { strcat(AST,s); }
   | LETTER<s> { strcat(AST,s); }
   | INITCAP<s> { strcat(AST,s); }
   | ALLCAPS<s> { strcat(AST,s); }
   | LETTERS<s> { strcat(AST,s); }
   | GENWORD<s> { strcat(AST,s); }
   | '.' { strcat(AST,"."); }
```

```
  )*
  ':'
  { strcat(AST,UP); }
;

// -----------------------------------------------------

defbody :
  %prelude { strcat(AST,DN); strcat(AST,"defbody"); }
  numeral ARY
  (  RELN rstr '.' relndef
   | FUNC fstr '.' funcdef
  )
  { strcat(AST,UP); }
;

// -----------------------------------------------------

infixdefbody :
  %prelude { strcat(AST,DN); strcat(AST,"infixdefbody"); }
  INFIX
  (  RELN rstr '.' infixrelndef
   | FUNC fstr '.' infixfuncdef
  )
  { strcat(AST,UP); }
;

// -----------------------------------------------------

relndef :
  %prelude { strcat(AST,DN); strcat(AST,"relndef"); }
  (  relndefunit '.'
   | IF formula THEN relndefunit '.'
     (IF formula THEN relndefunit '.')*
     (OTHERWISE relndefunit '.')?
  )
  { strcat(AST,UP); }
;
```

```
// ------------------------------------------------------

relndefunit :
  %prelude { strcat(AST,DN); strcat(AST,"relndefunit"); }
  rstr '[' varseq ']' IFF formula
  { strcat(AST,UP); }
;

// ------------------------------------------------------

infixrelndef :
  %prelude { strcat(AST,DN); strcat(AST,"infixrelndef"); }
  ( infixrelndefunit '.'
   | IF formula THEN infixrelndefunit '.'
     (IF formula THEN infixrelndefunit '.')*
     (OTHERWISE infixrelndefunit '.')?
  )
  { strcat(AST,UP); }
;

// ------------------------------------------------------

infixrelndefunit :
  %prelude { strcat(AST,DN); strcat(AST,"infixrelndefunit"); }
  var rstr var IFF formula
  { strcat(AST,UP); }
;

// ------------------------------------------------------

funcdef :
  %prelude { strcat(AST,DN); strcat(AST,"funcdef"); }
  ( funcdefunit '.'
   | IF formula THEN funcdefunit '.'
     (IF formula THEN funcdefunit '.')*
     (OTHERWISE funcdefunit '.')?
  )
```

```
  { strcat(AST,UP); }
;

// -----------------------------------------------------

funcdefunit :
  %prelude { strcat(AST,DN); strcat(AST,"funcdefunit"); }
  composite (SIMEQ term | UPARROW)
  { strcat(AST,UP); }
;

// -----------------------------------------------------

infixfuncdef :
  %prelude { strcat(AST,DN); strcat(AST,"infixfuncdef"); }
  ( infixfuncdefunit '.' precedence '.'
   | IF formula THEN infixfuncdefunit '.'
     (IF formula THEN infixfuncdefunit '.')*
     (OTHERWISE infixfuncdefunit '.')? precedence '.'
  )
  { strcat(AST,UP); }
;

// -----------------------------------------------------

infixfuncdefunit :
  %prelude { strcat(AST,DN); strcat(AST,"infixfuncdefunit"); }
  var fstr var (SIMEQ term | UPARROW)
  { strcat(AST,UP); }
;

// -----------------------------------------------------

precedence :
  %prelude { strcat(AST,DN); strcat(AST,"precedence"); }
  { strcat(AST,"$$"); }
  PRECEDENCE ('-' {strcat(AST,"-");})?
  (  '0' { strcat(AST,"0"); }
```

```
   | NZNUM<s> { strcat(AST,s); }
   | DIGITS<s> { strcat(AST,s); }
  )
  { strcat(AST,UP); }
;

// ---------------------- BRANCHES ----------------------

term :
  %prelude { strcat(AST,DN); strcat(AST,"term"); }
  bterm (infixfunc bterm)*
  { strcat(AST,UP); }
;

// ------------------------------------------------------

bterm :
  %prelude { strcat(AST,DN); strcat(AST,"bterm"); }
  aterm (evaluation)*
  { strcat(AST,UP); }
;

// ------------------------------------------------------

aterm :
  %prelude { strcat(AST,DN); strcat(AST,"aterm"); }
  ( var
  | numeral
  | composite
  | '(' term ')'
  | set
  | tuple
  | unique
  | lambda
  )
  { strcat(AST,UP); }
;
```

```
// ------------------------------------------------------

varseq :
  %prelude { strcat(AST,DN); strcat(AST,"varseq"); }
  var (',' var)*
  { strcat(AST,UP); }
;

// ------------------------------------------------------

composite :
  %prelude { strcat(AST,DN); strcat(AST,"composite"); }
  fstr (evaluation)?
  { strcat(AST,UP); }
;

// ------------------------------------------------------

set :
  %prelude { strcat(AST,DN); strcat(AST,"set"); }
  '{' term (  (',' termseq)? {strcat(AST,"$$");} '}'
            | (infixrel termseq)? ':' formula (  '}'
                                               | ',' varseq FIXED '}'
                                              )
           )
  { strcat(AST,UP); }
;

// ------------------------------------------------------

tuple :
  %prelude { strcat(AST,DN); strcat(AST,"tuple"); }
  '<' termseq '>'
  { strcat(AST,UP); }
;

// ------------------------------------------------------
```

```
unique :
  %prelude { strcat(AST,DN); strcat(AST,"unique"); }
  '(' '!' term (infixrel termseq)? ')' (  bformula
                                        | '(' formula ',' varseq FIXED ')'
                                       )
  { strcat(AST,UP); }
;

// -----------------------------------------------------

lambda :
  %prelude { strcat(AST,DN); strcat(AST,"lambda"); }
  '(' LAMBDA varseq (infixrel termseq)? (':' formula)? ')' bterm
  { strcat(AST,UP); }
;

// -----------------------------------------------------

evaluation :
  %prelude { strcat(AST,DN); strcat(AST,"evaluation"); }
  '(' termseq ')'
  { strcat(AST,UP); }
;

// -----------------------------------------------------

infixfunc :
  %prelude { strcat(AST,DN); strcat(AST,"infixfunc"); }
  (  fstr
   | INFIXFN '{' term '}'
  )
  { strcat(AST,UP); }
;

// -----------------------------------------------------

infixrel :
  %prelude { strcat(AST,DN); strcat(AST,"infixrel"); }
```

```
  (  rstr
   | '=' { strcat(AST,"$$="); }
   | NEQ { strcat(AST,"$$\\neq"); }
   | SIMEQ { strcat(AST,"$$\\simeq"); }
   | INFIXRL '{' term '}'
  )
  { strcat(AST,UP); }
;

// -----------------------------------------------------

termseq :
  %prelude { strcat(AST,DN); strcat(AST,"termseq"); }
  term (',' term)*
  { strcat(AST,UP); }
;

// -----------------------------------------------------

formula :
  %prelude { strcat(AST,DN); strcat(AST,"formula"); }
  bformula (lc bformula)*
  { strcat(AST,UP); }
;

// -----------------------------------------------------

lc :
  %prelude { strcat(AST,DN); strcat(AST,"lc"); }
  { strcat(AST,"$$"); }
  ( VEE { strcat(AST,"\\vee"); }
  | WEDGE { strcat(AST,"\\wedge"); }
  | THENSYM { strcat(AST,"\\rightarrow"); }
  | IFF { strcat(AST,"\\rightleftarrow"); }
  )
  { strcat(AST,UP); }
;
```

```
// ------------------------------------------------------

bformula :
  %prelude { strcat(AST,DN); strcat(AST,"bformula"); }
  ( BOT { strcat(AST,"$$\\bot"); }
  | TOP { strcat(AST,"$$\\top"); }
  | termformula
  | relation
  | '(' formula ')'
  | negation
  | onetrue
  | forall
  | exists
  | existsunique
  )
  { strcat(AST,UP); }
;

// ------------------------------------------------------

termformula :
  %prelude { strcat(AST,DN); strcat(AST,"termformula"); }
  term (  UPARROW { strcat(AST,"$$\\uparrow"); }
        | DOWN { strcat(AST,"$$\\downarrow"); }
        | {strcat(AST,"$$");} '[' termseq ']'
        | (',' termseq)? infixrel termseq (infixrel termseq)*
       )
  { strcat(AST,UP); }
;

// ------------------------------------------------------

relation :
  %prelude { strcat(AST,DN); strcat(AST,"relation"); }
  rstr '[' termseq ']'
  { strcat(AST,UP); }
;
```

```
// ------------------------------------------------------

negation :
  %prelude { strcat(AST,DN); strcat(AST,"negation"); }
  NEG bformula
  { strcat(AST,UP); }
;

// ------------------------------------------------------

onetrue :
  %prelude { strcat(AST,DN); strcat(AST,"onetrue"); }
  '!' '[' formulaseq ']'
  { strcat(AST,UP); }
;

// ------------------------------------------------------

formulaseq :
  %prelude { strcat(AST,DN); strcat(AST,"formulaseq"); }
  formula (',' formula)*
  { strcat(AST,UP); }
;

// ------------------------------------------------------

forall :
  %prelude { strcat(AST,DN); strcat(AST,"forall"); }
  '(' FORALL termseq
  (  ':' { strcat(AST,"$$"); } formula ')' bformula
   | (infixrel termseq)? ')'
     (  bformula
      | '(' formula ',' varseq FIXED ')'
     )
  )
  { strcat(AST,UP); }
;
```

```
// ------------------------------------------------------

exists :
  %prelude { strcat(AST,DN); strcat(AST,"exists"); }
  '(' EXISTS termseq
  (  ':' { strcat(AST,"$$"); } formula ')' bformula
   | (infixrel termseq)? ')'
     (  bformula
      | '(' formula ',' varseq FIXED ')'
     )
  )
  { strcat(AST,UP); }
;

// ------------------------------------------------------

existsunique :
  %prelude { strcat(AST,DN); strcat(AST,"existsunique"); }
  '(' EXISTS '!' termseq
  (  ':' { strcat(AST,"$$"); } formula ')' bformula
   | (infixrel termseq)? ')'
     (  bformula
      | '(' formula ',' varseq FIXED ')'
     )
  )
  { strcat(AST,UP); }
;

// --------------------- LEAVES -------------------------

var :
  %prelude { strcat(AST,DN); strcat(AST,"var"); }
  { strcat(AST,"$$"); }
  (DECORATION<s> { strcat(AST,s); })?
  ( FONT<s> { strcat(AST,s); }
    '{' { strcat(AST,"{"); }
    LETTER<s> { strcat(AST,s); }
    '}' { strcat(AST,"}"); }
```

```
  | LETTER<s> { strcat(AST,s); }
  | MISCLETTER<s> { strcat(AST,s); }
  | GREEK<s> { strcat(AST,s); }
  )
  (''' { strcat(AST,"'"); })*
  (  '_' '{' { strcat(AST,"_{"); }
     ( '0' { strcat(AST,"0"); }
      | NZNUM<s> { strcat(AST,s); }
      | DIGITS<s> { strcat(AST,s); }
     )
     '}' { strcat(AST,"}"); }
     ('*' { strcat(AST,"^*"); })?
   | '*' { strcat(AST,"^*"); }
     ('_' '{' { strcat(AST,"_{"); }
      ( '0' { strcat(AST,"0"); }
       | NZNUM<s> { strcat(AST,s); }
       | DIGITS<s> { strcat(AST,s); }
      )
      '}' { strcat(AST,"}"); }
     )?
  )?
  { strcat(AST,UP); }
;

// ------------------------------------------------------

fstr :
  %prelude { strcat(AST,DN); strcat(AST,"fstr"); }
  { strcat(AST,"$$"); }
  ( (  FUNCSYM<s> { strcat(AST,s); }
     | '-' { strcat(AST,"-"); }
     | INITCAP<s> { strcat(AST,s); }
     | FONT<s> {strcat(AST,s);}
       '{' {strcat(AST,"{");}
       INITCAP<s> {strcat(AST,s);}
       '}' {strcat(AST,"}");}
     | MATHBB '{' {strcat(AST,"\\mathbb{");}
       LETTER<s> {strcat(AST,s);}
```

```
      '}' {strcat(AST,"}");}
   )
   ( (''' { strcat(AST,"'"); })*
     (  '_' '{' { strcat(AST,"_{"); }
        chars (chars)*
        '}' { strcat(AST,"}"); }
        ('*' { strcat(AST,"^*"); })?
      | '*' { strcat(AST,"^*"); }
        ('_' '{' { strcat(AST,"_{"); }
         chars (chars)*
         '}' { strcat(AST,"}"); }
        )?
     )?
   )?
 | (  '0' { strcat(AST,"0"); }
    | NZNUM<s> { strcat(AST,s); }
    | GREEK<s> { strcat(AST,s); }
   )
   (  ''' { strcat(AST,"'"); }
      (''' { strcat(AST,"'"); })*
      (  '_' '{' { strcat(AST,"_{"); }
         chars (chars)*
         '}' { strcat(AST,"}"); }
         ('*' { strcat(AST,"^*"); })?
       | '*' { strcat(AST,"^*"); }
         ('_' '{' { strcat(AST,"_{"); }
          chars (chars)*
          '}' { strcat(AST,"}"); }
         )?
      )?
    | (  '_' '{' { strcat(AST,"_{"); }
         chars (chars)*
         '}' { strcat(AST,"}"); }
         ('*' { strcat(AST,"^*"); })?
       | '*' { strcat(AST,"^*"); }
         ('_' '{' { strcat(AST,"_{"); }
          chars (chars)*
          '}' { strcat(AST,"}"); }
```

```
          )?
       )
    )
  )
  { strcat(AST,UP); }
;

// -----------------------------------------------------

rstr :
  %prelude { strcat(AST,DN); strcat(AST,"rstr"); }
  { strcat(AST,"$$"); }
  (  RELNSYM<s> { strcat(AST,s); }
   | '<' { strcat(AST,"<"); }
   | '>' { strcat(AST,">"); }
   | ALLCAPS<s> { strcat(AST,s); }
   | FONT<s> {strcat(AST,s);}
       '{' {strcat(AST,"{");}
       ALLCAPS<s> {strcat(AST,s);}
       '}' {strcat(AST,"}");}
  )
  ( (''' { strcat(AST,"'"); })*
      (  '_' '{' { strcat(AST,"_{"); }
         chars (chars)*
         '}' { strcat(AST,"}"); }
         ('*' { strcat(AST,"^*"); })?
       | '*' { strcat(AST,"^*"); }
         ('_' '{' { strcat(AST,"_{"); }
          chars (chars)*
          '}' { strcat(AST,"}"); }
         )?
      )?
  )?
  { strcat(AST,UP); }
;

// -----------------------------------------------------
```

```
numeral :
  %prelude { strcat(AST,DN); strcat(AST,"numeral"); }
  (  '0' { strcat(AST,"$$0"); }
   | NZNUM<s> { strcat(AST,"$$"); strcat(AST,s); }
  )
  { strcat(AST,UP); }
;

// -----------------------------------------------------

chars :
    '0' { strcat(AST,"0"); }
  | NZNUM<s> { strcat(AST,s); }
  | DIGITS<s> { strcat(AST,s); }
  | LETTER<s> { strcat(AST,s); }
  | INITCAP<s> { strcat(AST,s); }
  | ALLCAPS<s> { strcat(AST,s); }
  | LETTERS<s> { strcat(AST,s); }
  | MISCLETTER<s> { strcat(AST,s); }
  | FONT<s> { strcat(AST,s); }
    '{' { strcat(AST,"{"); }
    (  LETTER<s> { strcat(AST,s); }
     | INITCAP<s> { strcat(AST,s); }
     | ALLCAPS<s> { strcat(AST,s); }
     | LETTERS<s> { strcat(AST,s); }
    )
    '}' { strcat(AST,"}"); }
  | MATHBB<s> { strcat(AST,s); }
    '{' { strcat(AST,"{"); }
    LETTER<s> { strcat(AST,s); }
    '}' { strcat(AST,"}"); }
  | GREEK<s> { strcat(AST,s); }
  | FUNCSYM<s> { strcat(AST,s); }
  | RELNSYM<s> { strcat(AST,s); }
;
```

Appendix C

DAG depth and size data

In the tables below, find all the DAG depth and size measurements for the definitions appearing in Appendices F and G.

C.1 Foundations – Suppes

Suppes Chapter 2

Definiendum	Depth	Size
Δ_0:infix	2	3
$\times$:infix	2	2

Suppes Chapter 3

Definiendum	Depth	Size	Definiendum	Depth	Size
BR:1	2	2	TR:1	2	2
Dom:1	3	3	Rng:1	3	3
Fld:1	4	6	Cnv:1	3	3
$\circ$:infix	3	3	$\vert$:infix	4	6
$\vert_{\texttt{Rng}}$:infix	5	7	RFX:2	3	3
IRFX:2	3	3	SYM:2	3	3
ASYM:2	3	3	ANSYM:2	3	3
TRAN:2	3	3	CONN:2	3	3
SCONN:2	3	3	RFX:1	5	8
IRFX:1	5	8	SYM:1	4	5

Suppes Chapter 3, continued

Definiendum	Depth	Size	Definiendum	Depth	Size
ASYM:1	4	5	ANSYM:1	4	5
TRAN:1	4	5	CONN:1	4	5
SCONN:1	4	5	Id:1	2	2
QORD:2	4	5	PORD:2	4	6
SORD:2	4	6	SPORD:2	4	5
SSORD:2	4	6	QORD:1	5	10
PORD:1	5	11	SORD:1	5	11
SPORD:1	5	10	SSORD:1	5	11
MELT:3	3	3	FELT:3	3	3
WO:2	4	7	ISUC:3	3	3
LELT:3	3	3	SECT:3	6	10
Seg:3	3	3	LB:3	3	3
INF:3	4	4	UB:3	3	3
SUP:3	4	4	EQUIV:1	6	13
EQUIV:2	7	14	Coset:2	7	14
PART:2	2	4	PART:1	3	5
FINER:2	4	7	Part:1	8	15
Reln:1	4	7	FCN:1	2	2
MONO:1	4	5	FCN:3	4	7
SURJ:3	4	6	MONO:3	5	9
BIJ:3	5	8	Maps:2	5	8

Suppes Chapter 4

Definiendum	Depth	Size	Definiendum	Depth	Size
$\approx_c$:infix	6	9	$\leq_c$:infix	7	11
$<_c$:infix	8	12	MNEL:2	1	1
MXEL:2	1	1	FIN:1	3	5
DFIN:1	7	11			

Suppes Chapter 5

Definiendum	Depth	Size	Definiendum	Depth	Size
TRANS:1	1	1	ECONN:1	1	1
ORD:1	2	3	Eps:1	2	2
$<_{\mathtt{O}}$:infix	3	4	$\leq_{\mathtt{O}}$:infix	3	4
$>_{\mathtt{O}}$:infix	3	4	$\geq_{\mathtt{O}}$:infix	3	4
Suc:1	4	5	NAT:1	5	13
ω:0	6	14	$\mathbb{N}$:0	7	15
$0_{\mathtt{N}}$:0	2	2	$1_{\mathtt{N}}$:0	2	2
$2_{\mathtt{N}}$:0	5	8	$3_{\mathtt{N}}$:0	6	9
$4_{\mathtt{N}}$:0	7	10	$5_{\mathtt{N}}$:0	8	11
$6_{\mathtt{N}}$:0	9	12	$7_{\mathtt{N}}$:0	10	13
$8_{\mathtt{N}}$:0	11	14	$9_{\mathtt{N}}$:0	12	15
$10_{\mathtt{N}}$:0	13	16	Addnat:0	7	18
$+_{\mathtt{N}}$:infix	8	19	Mulnat:0	9	21
$\cdot_{\mathtt{N}}$:infix	10	22	Expnat:0	11	23
$\mathtt{Exp}_{\mathtt{N}}$:infix	12	24	INF:1	4	6
DEN:1	7	21	DINF:1	8	12
CNTBL:1	7	20	UNCNTBL:1	8	21

Suppes Chapter 6

Definiendum	Depth	Size	Definiendum	Depth	Size
/:infix	7	16	Fr:0	8	17
$\equiv_{\mathtt{Fr}}$:infix	11	24	$<_{\mathtt{Fr}}$:infix	11	25
$>_{\mathtt{Fr}}$:infix	12	26	$\leq_{\mathtt{Fr}}$:infix	12	27
$\geq_{\mathtt{Fr}}$:infix	13	28	$+_{\mathtt{Fr}}$:infix	11	24
$\cdot_{\mathtt{Fr}}$:infix	11	24	Nra:0	12	38
$\mathtt{Incl}_{\mathtt{FrNra}}$:1	12	38	$<_{\mathtt{Nra}}$:infix	13	41
$>_{\mathtt{Nra}}$:infix	13	42	$\leq_{\mathtt{Nra}}$:infix	13	42
$\geq_{\mathtt{Nra}}$:infix	14	43	$+_{\mathtt{Nra}}$:infix	13	40
$\cdot_{\mathtt{Nra}}$:infix	13	40	$0_{\mathtt{Nra}}$:0	12	37
$1_{\mathtt{Nra}}$:0	12	37	$\equiv_{\mathtt{SUB}}$:infix	14	41
$<_{\mathtt{SUB}}$:infix	14	44	$+_{\mathtt{SUB}}$:infix	14	41
$\cdot_{\mathtt{SUB}}$:infix	14	43	Ra:0	15	43
$\mathtt{Incl}_{\mathtt{NraRa}}$:1	15	43	$\mathtt{Incl}_{\mathtt{FrRa}}$:1	16	45

Suppes Chapter 6, continued

Definiendum	Depth	Size	Definiendum	Depth	Size
$\mathbb{Q}$:0	16	44	$<_{\mathtt{Ra}}$:infix	16	48
$+_{\mathtt{Ra}}$:infix	16	45	$\cdot_{\mathtt{Ra}}$:infix	16	47
$\mathtt{0}_{\mathtt{Ra}}$:0	15	43	$\mathtt{1}_{\mathtt{Ra}}$:0	15	44
$>_{\mathtt{Ra}}$:infix	17	49	$\leq_{\mathtt{Ra}}$:infix	17	49
$\geq_{\mathtt{Ra}}$:infix	18	50	$-_{\mathtt{Ra}}$:infix	17	46
$\mathtt{Av}_{\mathtt{Ra}}$:1	19	56	$\mathtt{Nat}_{\mathtt{Ra}}$:0	18	57
$\mathbb{N}_{\mathtt{Ra}}$:0	19	58	$\mathtt{Int}_{\mathtt{Ra}}$:0	19	58
$\mathbb{Z}_{\mathtt{Ra}}$:0	20	59	$\mathtt{Seq}_{\mathtt{Ra}}$:0	16	47
$+_{\mathtt{SeqRa}}$:infix	17	50	$\cdot_{\mathtt{SeqRa}}$:infix	17	52
$<_{\mathtt{N}}$:infix	7	16	$>_{\mathtt{N}}$:infix	7	16
$\leq_{\mathtt{N}}$:infix	8	17	$\geq_{\mathtt{N}}$:infix	8	17
$\mathtt{Cseq}_{\mathtt{Ra}}$:0	20	63	$\equiv_{\mathtt{CseqRa}}$:infix	20	59
$<_{\mathtt{CseqRa}}$:infix	21	64	$\mathbb{R}$:0	21	65
$<_{\mathtt{R}}$:infix	22	67	$>_{\mathtt{R}}$:infix	23	68
$\leq_{\mathtt{R}}$:infix	23	68	$\geq_{\mathtt{R}}$:infix	24	69
$+_{\mathtt{R}}$:infix	22	67	$-_{\mathtt{R}}$:infix	23	68
$\cdot_{\mathtt{R}}$:infix	22	71	$\mathtt{0}_{\mathtt{R}}$:0	22	66
$\mathtt{1}_{\mathtt{R}}$:0	22	68	$\mathtt{Av}_{\mathtt{R}}$:1	25	74
$\mathtt{Id}_{\mathtt{R}}$:1	22	66	$\mathtt{Incl}_{\mathtt{NraR}}$:1	23	68
$\mathtt{Incl}_{\mathtt{FrR}}$:1	23	70	$\mathtt{Ra}_{\mathtt{R}}$:0	23	67
$\mathtt{Seq}_{\mathtt{R}}$:0	22	66	$\mathtt{Cseq}_{\mathtt{R}}$:0	26	76
Lim:1	26	76	$\mathtt{UB}_{\mathtt{R}}$:2	24	69
$\mathtt{Min}_{\mathtt{R}}$:1	23	68	$\mathtt{Max}_{\mathtt{R}}$:1	23	68
$\mathtt{Lub}_{\mathtt{R}}$:1	25	71	$\mathtt{Finitesumset}_{\mathtt{R}}$:1	23	71
$\mathtt{Finitesum}_{\mathtt{R}}$:1	24	72	$\mathtt{Sqrt}_{\mathtt{R}}$:1	25	77
$\mathtt{Sup}_{\mathtt{R}}$:1	23	70	$\mathtt{Inf}_{\mathtt{R}}$:1	23	70

C.2 Topology – Munkres

Munkres Chapter 12

Definiendum	Depth	Size	Definiendum	Depth	Size
TOPOLOGY:2	3	6	TOPSP:2	4	7
OPENSET:3	5	8	FINERTOP:3	5	9
STRFINERTOP:3	5	9	COARSERTOP:3	5	8
STRCOARSERTOP:3	5	8			

Munkres Chapter 13

Definiendum	Depth	Size	Definiendum	Depth	Size
TOPBASIS:2	3	4	Basisgentop:2	4	5
Stdrealtopbasis:0	23	68	Stdrealtop:0	24	73
Lowerlimitrealtop:0	24	73	Krealtop:0	24	80
TOPSUBBASIS:2	3	4	Subbasisgentop:2	8	15

Munkres Chapter 14

Definiendum	Depth	Size	Definiendum	Depth	Size
Oointerval:4	5	7	OOINTERVAL:3	6	9
Ocinterval:4	5	7	OCINTERVAL:3	6	9
Cointerval:4	5	7	COINTERVAL:3	6	9
Ccinterval:4	5	7	CCINTERVAL:3	6	9
Ordertopbasis:2	6	12	Ordertop:2	7	18
Oplusray:3	5	7	Ominusray:3	5	7
Cplusray:3	5	7	Cminusray:3	5	7

Munkres Chapter 15

Definiendum	Depth	Size	Definiendum	Depth	Size
Prodtopbasis:4	5	10	Prodtop:4	6	13
π_1:2	1	1	π_2:2	1	1
π_1:1	2	2	π_2:1	2	2

Munkres Chapter 16

Definiendum	Depth	Size	Definiendum	Depth	Size
Subspacetop:3	5	8	SUBSPACE:4	6	9
Dictionaryorder:4	5	7	Orderedsquare:0	23	87
CONVEX:3	6	9			

Munkres Chapter 17

Definiendum	Depth	Size	Definiendum	Depth	Size
CLOSEDSET:3	6	10	Interior:1	5	8
Closure:3	7	12	INTERSECTS:2	2	3
DISJOINT:2	2	3	NBHD:4	5	8
LIMITPT:4	6	11	TOPCONV:3	8	29
HAUSDORFF:2	5	9	TOPLIMIT:3	9	32

Munkres Chapter 18

Definiendum	Depth	Size	Definiendum	Depth	Size
CONTINUOUS:5	6	19	CONTAT:6	6	19
HOMEOMORPHISM:5	7	22	HOMEOMORPHIC:4	8	23
TOPIMBED:5	8	25			

Munkres Chapter 19

Definiendum	Depth	Size	Definiendum	Depth	Size
TUPLE:3	4	5	Cartesprod:2	5	6
Cartespow:2	6	7	Boxtopbasis:3	6	14
Boxtop:3	7	17	Projection:3	6	7
Prodtopsubbasis:3	7	20	Productspace:3	9	27

Munkres Chapter 20

Definiendum	Depth	Size	Definiendum	Depth	Size
METRIC:2	25	73	Ball:4	26	74
Metrictopbasis:2	27	75	Metrictop:2	28	80
METRIZABLE:2	29	84	METRICSPACE:3	29	84
BOUNDED:3	30	86	Diam:1	31	89
Stdbddmetric:2	30	89	Rnnorm:2	26	87
Rneuclideanmetric:3	27	89	Rnsqmetric:3	26	79
Uniformmetric:3	31	110			

Munkres Chapter 21

Definiendum	Depth	Size	Definiendum	Depth	Size
CNTBLBASISAT:3	8	26	FIRSTCNTBL:2	9	27
UNIFCONV:7	30	86			

Munkres Chapter 22

Definiendum	Depth	Size	Definiendum	Depth	Size
QUOTIENTMAP:5	6	18	SATURATED:5	6	19
OPENMAP:5	6	17	CLOSEDMAP:5	7	20
Quotienttop:3	7	19	QUOTIENTSPACE:2	8	21

Munkres Chapter 23

Definiendum	Depth	Size
SEPARATION:4	5	10
CONNECTED:2	6	11
TOTALLYDISCONNECTED:2	7	23

Munkres Chapter 24

Definiendum	Depth	Size	Definiendum	Depth	Size
LINEARCONTINUUM:2	9	21	PATH:5	25	84
PATHCONNECTED:2	26	85	Rnunitball:1	27	93
Puncturedeuclideanspace:1	25	84	Unitsphere:1	27	92

Munkres Chapter 25

Definiendum	Depth	Size	Definiendum	Depth	Size
Ccequiv:2	7	28	$\sim_{CC}$:infix	8	29
Components:2	9	40	Pcequiv:2	26	89
$\sim_{PC}$:infix	27	90	Pathcomponents:2	27	91
LOCALCONNAT:3	7	14	LOCALCONN:2	8	15
LOCALPATHCONNAT:3	27	87	LOCALPATHCONN:2	28	88
WEAKLOCALCONNAT:3	7	14			

Munkres Chapter 26

Definiendum	Depth	Size	Definiendum	Depth	Size
COVER:2	3	4	OPENCOVER:3	5	9
COMPACT:2	8	20	COVER:3	3	5
FINITEINTERSPROP:1	8	14			

Munkres Chapter 27

Definiendum	Depth	Size	Definiendum	Depth	Size
Distance:4	30	88	LEBESGUENUM:4	32	97
UNIFORMCONT:5	30	85	ISOLATEDPT:3	5	8

Munkres Chapter 28

Definiendum	Depth	Size	Definiendum	Depth	Size
LIMITPTCPT:2	7	15	SUBSEQ:2	7	22
SEQCPT:2	9	33	CNTBLCPT:2	8	30

Munkres Chapter 29

Definiendum	Depth	Size	Definiendum	Depth	Size
LOCCPTAT:3	9	23	LOCCPT:2	10	24
COMPACTIFICATION:4	9	30	Oneptcompactification:2	10	32

Munkres Chapter 30

Definiendum	Depth	Size	Definiendum	Depth	Size
SECONDCNTBL:2	8	28	DENSE:3	8	13
LINDELOF:2	8	28	SEPARABLESPACE:2	9	32
Sorgenfreyplane:0	25	74	GDELTA:3	8	29

Munkres Chapter 31

Definiendum	Depth	Size
REGULARSP:2	7	12
NORMALSP:2	7	12

Munkres Chapter 32

Definiendum	Depth	Size
COMPLETELYNORMALSP:2	8	14

Munkres Chapter 33

Definiendum	Depth	Size
SEPBYCONTFUNC:4	25	88
COMPLETELYREGULARSP:2	26	92
PERFECTLYNORMALSP:2	9	35

Munkres Chapter 34

Definiendum	Depth	Size
LOCALLYMETRIZABLE:2	30	87

Munkres Chapter 35

Definiendum	Depth	Size
EXTENDS:infix	6	9
UNIVEXTPROP:2	8	27

Munkres Chapter 36

Definiendum	Depth	Size	Definiendum	Depth	Size
Manifold:1	25	94	TOPCURVE:2	26	95
TOPSURFACE:2	26	96	Support:1	23	79
PARTITIONOFUNITY:4	25	95	POUDOMBY:5	26	104
POINTFINFAM:3	8	15			

Munkres Chapter 37

Definiendum	Depth	Size
CNTBLINTERSPROP:1	8	22

Munkres Chapter 38

Definiendum	Depth	Size
EQUIVCOMPACTIFICATION:6	10	38
Imbedinducedcptfcation:1	10	41
Stonecech:2	27	107

Appendix D

Quantifier depth data

We gathered eight data points for each definition in the database. In the following tables, we use *alpha* for a definition's translation into DZFC, $\mathscr{E}(\alpha, \varnothing)$ for the total expansion thereof, and $\mathscr{E}(\alpha, B_0)$ for the partial expansion, as explained in Section 4.2.

D.1 Foundations – Suppes

Suppes Chapter 2

	Nesting				**Alternating Nesting**			
Definiendum	LPT	α	$\mathscr{E}(\alpha, \varnothing)$	$\mathscr{E}(\alpha, B_0)$	LPT	α	$\mathscr{E}(\alpha, \varnothing)$	$\mathscr{E}(\alpha, B_0)$
Δ_0:infix	0	0	4	0	0	0	4	0
$\times$:infix	0	2	4	2	0	2	3	2

Suppes Chapter 3

	Nesting				**Alternating Nesting**			
Definiendum	LPT	α	$\mathscr{E}(\alpha, \varnothing)$	$\mathscr{E}(\alpha, B_0)$	LPT	α	$\mathscr{E}(\alpha, \varnothing)$	$\mathscr{E}(\alpha, B_0)$
BR:1	2	2	4	2	2	2	3	2
TR:1	2	2	6	2	2	2	3	2
Dom:1	1	3	5	3	1	2	3	2
Rng:1	1	3	5	3	1	2	3	2
Fld:1	0	0	7	3	0	0	5	2

Suppes Chapter 3 continued

	Nesting				Alternating Nesting			
Definiendum	LPT	α	$\mathscr{E}(\alpha, \varnothing)$	$\mathscr{E}(\alpha, B_0)$	LPT	α	$\mathscr{E}(\alpha, \varnothing)$	$\mathscr{E}(\alpha, B_0)$
Cnv:1	0	2	4	2	0	2	3	2
$\circ$:infix	1	3	5	3	1	2	3	2
$\vert$:infix	0	0	11	5	0	0	8	4
$\vert_{\texttt{Rng}}$:infix	0	0	16	8	0	0	11	6
RFX:2	1	1	4	2	1	1	3	2
IRFX:2	1	1	4	2	1	1	3	2
SYM:2	1	1	4	2	1	1	3	2
ASYM:2	1	1	4	2	1	1	3	2
ANSYM:2	1	1	4	2	1	1	3	2
TRAN:2	1	1	4	2	1	1	3	2
CONN:2	1	1	4	2	1	1	3	2
SCONN:2	1	1	4	2	1	1	3	2
RFX:1	0	0	11	4	0	0	8	3
IRFX:1	0	0	11	4	0	0	8	3
SYM:1	0	0	9	4	0	0	6	3
ASYM:1	0	0	9	4	0	0	6	3
ANSYM:1	0	0	9	4	0	0	6	3
TRAN:1	0	0	9	4	0	0	6	3
CONN:1	0	0	9	4	0	0	6	3
SCONN:1	0	0	9	4	0	0	6	3
Id:1	0	2	4	2	0	2	3	2
QORD:2	0	0	4	2	0	0	3	2
PORD:2	0	0	4	2	0	0	3	2
SORD:2	0	0	4	2	0	0	3	2
SPORD:2	0	0	4	2	0	0	3	2
SSORD:2	0	0	4	2	0	0	3	2
QORD:1	0	0	11	4	0	0	8	3
PORD:1	0	0	11	4	0	0	8	3
SORD:1	0	0	11	4	0	0	8	3
SPORD:1	0	0	11	4	0	0	8	3
SSORD:1	0	0	11	4	0	0	8	3
MELT:3	1	1	4	2	1	1	3	2
FELT:3	1	1	4	2	1	1	3	2
WO:2	2	2	6	4	2	2	5	4
ISUC:3	1	1	4	2	1	1	3	2
LELT:3	1	1	4	2	1	1	3	2
SECT:3	0	0	23	10	0	0	16	8
Seg:3	0	2	4	2	0	2	3	2
LB:3	1	1	4	2	1	1	3	2

Suppes Chapter 3 continued

	Nesting				Alternating Nesting			
Definiendum	LPT	α	$\mathscr{E}(\alpha, \varnothing)$	$\mathscr{E}(\alpha, B_0)$	LPT	α	$\mathscr{E}(\alpha, \varnothing)$	$\mathscr{E}(\alpha, B_0)$
INF:3	1	1	5	3	1	1	3	2
UB:3	1	1	4	2	1	1	3	2
SUP:3	1	1	5	3	1	1	3	2
EQUIV:1	0	0	11	4	0	0	8	3
EQUIV:2	0	0	11	4	0	0	8	3
Coset:2	0	2	11	4	0	2	8	3
PART:2	1	1	3	1	1	1	3	1
PART:1	1	1	4	2	1	1	4	2
FINER:2	2	2	4	2	2	2	4	2
Part:1	0	2	13	6	0	2	10	5
Reln:1	1	3	4	3	1	2	4	2
FCN:1	0	2	4	2	0	2	3	2
MONO:1	0	0	8	4	0	0	6	4
FCN:3	0	0	6	3	0	0	3	2
SURJ:3	0	0	5	3	0	0	3	2
MONO:3	0	0	8	4	0	0	6	4
BIJ:3	0	0	8	4	0	0	6	4
Maps:2	0	2	8	5	0	2	5	4

Suppes Chapter 4

	Nesting				Alternating Nesting			
Definiendum	LPT	α	$\mathscr{E}(\alpha, \varnothing)$	$\mathscr{E}(\alpha, B_0)$	LPT	α	$\mathscr{E}(\alpha, \varnothing)$	$\mathscr{E}(\alpha, B_0)$
$\approx_c$:infix	1	1	9	5	1	1	7	5
$\leq_c$:infix	1	1	10	6	1	1	7	5
$<_c$:infix	0	0	10	6	0	0	7	5
MNEL:2	1	1	1	1	1	1	1	1
MXEL:2	1	1	1	1	1	1	1	1
FIN:1	2	2	5	3	2	2	4	3
DFIN:1	1	1	10	6	1	1	8	6

Suppes Chapter 5

	Nesting				Alternating Nesting			
Definiendum	LPT	α	$\mathscr{E}(\alpha, \varnothing)$	$\mathscr{E}(\alpha, B_0)$	LPT	α	$\mathscr{E}(\alpha, \varnothing)$	$\mathscr{E}(\alpha, B_0)$
TRANS:1	2	2	2	2	1	1	1	1
ECONN:1	1	1	1	1	1	1	1	1
ORD:1	0	0	2	2	0	0	1	1
Eps:1	0	2	4	2	0	2	3	2
$<_0$:infix	0	0	2	2	0	0	1	1
$\leq_0$:infix	0	0	2	2	0	0	1	1
$>_0$:infix	0	0	2	2	0	0	1	1
$\geq_0$:infix	0	0	2	2	0	0	1	1
Suc:1	0	2	4	4	0	2	3	3
NAT:1	0	0	14	7	0	0	11	7
ω:0	0	2	16	9	0	2	13	9
$\mathbb{N}$:0	0	0	16	9	0	0	13	9
0_N:0	0	0	2	0	0	0	2	0
1_N:0	0	1	3	1	0	1	3	1
2_N:0	0	0	7	5	0	0	6	4
3_N:0	0	0	11	9	0	0	9	7
4_N:0	0	0	15	13	0	0	12	10
5_N:0	0	0	19	17	0	0	15	13
6_N:0	0	0	23	21	0	0	18	16
7_N:0	0	0	27	25	0	0	21	19
8_N:0	0	0	31	29	0	0	24	22
9_N:0	0	0	35	33	0	0	27	25
10_N:0	0	0	39	37	0	0	30	28
Addnat:0	1	2	18	11	1	2	14	10
$+_N$:infix	0	1	19	12	0	1	14	10
Mulnat:0	1	2	25	14	1	2	17	12
$\cdot_N$:infix	0	1	26	15	0	1	17	12
Expnat:0	1	2	28	17	1	2	19	14
Exp_N:infix	0	1	29	18	0	1	19	14
INF:1	0	0	5	3	0	0	4	3
DEN:1	0	0	25	14	0	0	20	14
DINF:1	0	0	10	6	0	0	8	6
CNTBL:1	1	1	25	14	1	1	20	14
UNCNTBL:1	0	0	25	14	0	0	20	14

Suppes Chapter 6

	Nesting				**Alternating Nesting**			
Definiendum	LPT	α	$\mathscr{E}(\alpha, \varnothing)$	$\mathscr{E}(\alpha, B_0)$	LPT	α	$\mathscr{E}(\alpha, \varnothing)$	$\mathscr{E}(\alpha, B_0)$
/:infix	0	0	16	9	0	0	13	9
$\mathtt{Fr}$:0	0	2	18	11	0	2	15	11
$\equiv_{\mathtt{Fr}}$:infix	1	1	27	16	1	1	17	12
$<_{\mathtt{Fr}}$:infix	1	1	29	18	1	1	19	14
$>_{\mathtt{Fr}}$:infix	0	0	29	18	0	0	19	14
$\leq_{\mathtt{Fr}}$:infix	0	0	29	18	0	0	19	14
$\geq_{\mathtt{Fr}}$:infix	0	0	29	18	0	0	19	14
$+_{\mathtt{Fr}}$:infix	1	2	38	19	1	1	21	14
$\cdot_{\mathtt{Fr}}$:infix	1	2	28	17	1	1	17	12
$\mathtt{Nra}$:0	0	4	42	24	0	4	28	18
$\mathtt{Incl}_{\mathtt{FrNra}}$:1	0	2	40	22	0	2	26	16
$<_{\mathtt{Nra}}$:infix	1	1	42	24	1	1	28	18
$>_{\mathtt{Nra}}$:infix	1	1	42	24	1	1	28	18
$\leq_{\mathtt{Nra}}$:infix	1	1	42	24	1	1	28	18
$\geq_{\mathtt{Nra}}$:infix	1	1	42	24	1	1	28	18
$+_{\mathtt{Nra}}$:infix	1	2	46	25	1	1	29	19
$\cdot_{\mathtt{Nra}}$:infix	1	2	43	25	1	1	29	19
$0_{\mathtt{Nra}}$:0	0	2	40	22	0	2	26	16
$1_{\mathtt{Nra}}$:0	0	2	40	22	0	2	26	16
$\equiv_{\mathtt{SUB}}$:infix	1	1	47	26	1	1	29	19
$<_{\mathtt{SUB}}$:infix	1	1	56	31	1	1	33	22
$+_{\mathtt{SUB}}$:infix	1	2	73	36	1	1	37	23
$\cdot_{\mathtt{SUB}}$:infix	1	2	95	45	1	1	45	27
$\mathtt{Ra}$:0	0	4	62	34	0	4	40	25
$\mathtt{Incl}_{\mathtt{NraRa}}$:1	0	2	60	32	0	2	38	23
$\mathtt{Incl}_{\mathtt{FrRa}}$:1	0	0	79	38	0	0	43	24
$\mathbb{Q}$:0	0	0	62	34	0	0	40	25
$<_{\mathtt{Ra}}$:infix	1	1	63	35	1	1	41	26
$+_{\mathtt{Ra}}$:infix	1	2	75	38	1	1	41	26
$\cdot_{\mathtt{Ra}}$:infix	1	2	97	47	1	1	45	27
$0_{\mathtt{Ra}}$:0	0	2	60	32	0	2	38	23
$1_{\mathtt{Ra}}$:0	0	2	60	32	0	2	38	23
$>_{\mathtt{Ra}}$:infix	0	0	63	35	0	0	41	26
$\leq_{\mathtt{Ra}}$:infix	0	0	63	35	0	0	41	26
$\geq_{\mathtt{Ra}}$:infix	0	0	63	35	0	0	41	26
$-_{\mathtt{Ra}}$:infix	0	1	76	39	0	1	41	26
$\mathtt{Av}_{\mathtt{Ra}}$:1	0	1	116	56	0	1	54	31
$\mathtt{Nat}_{\mathtt{Ra}}$:0	1	2	117	57	1	2	56	33
$\mathbb{N}_{\mathtt{Ra}}$:0	0	0	117	57	0	0	56	33
$\mathtt{Int}_{\mathtt{Ra}}$:0	1	2	119	59	1	2	58	35

Suppes Chapter 6 continued

	Nesting				Alternating Nesting			
Definiendum	LPT	α	$\mathscr{E}(\alpha, \varnothing)$	$\mathscr{E}(\alpha, B_0)$	LPT	α	$\mathscr{E}(\alpha, \varnothing)$	$\mathscr{E}(\alpha, B_0)$
$\mathbb{Z}_{\mathtt{Ra}}$:0	0	0	119	59	0	0	58	35
$\mathtt{Seq}_{\mathtt{Ra}}$:0	0	0	70	39	0	0	45	29
$+_{\mathtt{SeqRa}}$:infix	1	2	77	40	1	2	46	30
$\cdot_{\mathtt{SeqRa}}$:infix	1	2	99	49	1	2	47	30
$<_{\mathtt{N}}$:infix	0	0	16	9	0	0	13	9
$>_{\mathtt{N}}$:infix	0	0	16	9	0	0	13	9
$\leq_{\mathtt{N}}$:infix	0	0	16	9	0	0	13	9
$\geq_{\mathtt{N}}$:infix	0	0	16	9	0	0	13	9
$\mathtt{Cseq}_{\mathtt{Ra}}$:0	3	5	173	84	3	5	74	46
$\equiv_{\mathtt{CseqRa}}$:infix	3	3	171	82	3	3	72	44
$<_{\mathtt{CseqRa}}$:infix	3	3	173	84	2	2	74	46
$\mathbb{R}$:0	0	4	186	90	0	4	84	51
$<_{\mathtt{R}}$:infix	1	1	187	91	1	1	85	52
$>_{\mathtt{R}}$:infix	0	0	187	91	0	0	85	52
$\leq_{\mathtt{R}}$:infix	0	0	187	91	0	0	85	52
$\geq_{\mathtt{R}}$:infix	0	0	187	91	0	0	85	52
$+_{\mathtt{R}}$:infix	1	2	231	108	1	1	95	57
$-_{\mathtt{R}}$:infix	0	1	232	109	0	1	95	57
$\cdot_{\mathtt{R}}$:infix	1	2	253	117	1	1	96	57
$0_{\mathtt{R}}$:0	2	3	187	91	2	2	85	52
$1_{\mathtt{R}}$:0	2	3	187	91	2	2	85	52
$\mathtt{Av}_{\mathtt{R}}$:1	0	1	401	185	0	1	153	89
$\mathtt{Id}_{\mathtt{R}}$:1	2	3	187	91	2	2	85	52
$\mathtt{Incl}_{\mathtt{NraR}}$:1	0	0	187	91	0	0	85	52
$\mathtt{Incl}_{\mathtt{FrR}}$:1	0	0	187	91	0	0	85	52
$\mathtt{Ra}_{\mathtt{R}}$:0	0	2	189	93	0	2	86	53
$\mathtt{Seq}_{\mathtt{R}}$:0	0	0	194	95	0	0	89	55
$\mathtt{Cseq}_{\mathtt{R}}$:0	3	5	547	250	3	5	199	116
Lim:1	3	4	546	249	3	4	198	115
$\mathtt{UB}_{\mathtt{R}}$:2	1	1	188	92	1	1	86	53
$\mathtt{Min}_{\mathtt{R}}$:1	1	2	189	93	1	2	87	54
$\mathtt{Max}_{\mathtt{R}}$:1	1	2	189	93	1	2	87	54
$\mathtt{Lub}_{\mathtt{R}}$:1	0	2	288	138	0	2	121	74
$\mathtt{Finitesumset}_{\mathtt{R}}$:1	2	2	233	110	2	2	97	59
$\mathtt{Finitesum}_{\mathtt{R}}$:1	1	1	234	111	1	1	97	59
$\mathtt{Sqrt}_{\mathtt{R}}$:1	0	1	284	134	0	1	116	69
$\mathtt{Sup}_{\mathtt{R}}$:1	0	3	195	96	0	3	90	55
$\mathtt{Inf}_{\mathtt{R}}$:1	0	3	195	96	0	3	90	55

D.2 Topology – Munkres

Munkres Chapter 12

	Nesting				Alternating Nesting			
Definiendum	LPT	α	$\mathscr{E}(\alpha, \varnothing)$	$\mathscr{E}(\alpha, B_0)$	LPT	α	$\mathscr{E}(\alpha, \varnothing)$	$\mathscr{E}(\alpha, B_0)$
TOPOLOGY:2	1	1	4	1	1	1	3	1
TOPSP:2	0	0	4	1	0	0	3	1
OPENSET:3	0	0	4	1	0	0	3	1
FINERTOP:3	0	0	4	1	0	0	3	1
STRFINERTOP:3	0	0	4	1	0	0	3	1
COARSERTOP:3	0	0	4	1	0	0	3	1
STRCOARSERTOP:3	0	0	4	1	0	0	3	1

Munkres Chapter 13

	Nesting				Alternating Nesting			
Definiendum	LPT	α	$\mathscr{E}(\alpha, \varnothing)$	$\mathscr{E}(\alpha, B_0)$	LPT	α	$\mathscr{E}(\alpha, \varnothing)$	$\mathscr{E}(\alpha, B_0)$
TOPBASIS:2	3	3	6	3	2	2	4	2
Basisgentop:2	3	4	6	4	2	3	4	3
Stdrealtopbasis:0	1	5	192	96	1	4	88	55
Stdrealtop:0	0	0	198	100	0	0	92	58
Lowerlimitrealtop:0	1	5	198	100	1	4	92	58
Krealtop:0	1	5	203	103	1	4	96	60
TOPSUBBASIS:2	0	0	4	0	0	0	3	0
Subbasisgentop:2	2	5	14	10	2	4	10	8

Munkres Chapter 14

	Nesting				Alternating Nesting			
Definiendum	LPT	α	$\mathscr{E}(\alpha, \varnothing)$	$\mathscr{E}(\alpha, B_0)$	LPT	α	$\mathscr{E}(\alpha, \varnothing)$	$\mathscr{E}(\alpha, B_0)$
`Oointerval`:4	0	2	4	2	0	2	3	2
`OOINTERVAL`:3	1	1	5	3	1	1	4	3
`Ocinterval`:4	0	2	4	2	0	2	3	2
`OCINTERVAL`:3	1	1	5	3	1	1	4	3
`Cointerval`:4	0	2	4	2	0	2	3	2
`COINTERVAL`:3	1	1	5	3	1	1	4	3
`Ccinterval`:4	0	2	4	2	0	2	3	2
`CCINTERVAL`:3	1	1	5	3	1	1	4	3
`Ordertopbasis`:2	1	3	7	5	1	2	5	4
`Ordertop`:2	0	0	13	9	0	0	9	7
`Oplusray`:3	0	2	4	2	0	2	3	2
`Ominusray`:3	0	2	4	2	0	2	3	2
`Cplusray`:3	0	2	4	2	0	2	3	2
`Cminusray`:3	0	2	4	2	0	2	3	2

Munkres Chapter 15

	Nesting				Alternating Nesting			
Definiendum	LPT	α	$\mathscr{E}(\alpha, \varnothing)$	$\mathscr{E}(\alpha, B_0)$	LPT	α	$\mathscr{E}(\alpha, \varnothing)$	$\mathscr{E}(\alpha, B_0)$
`Prodtopbasis`:4	0	2	6	4	0	2	5	4
`Prodtop`:4	0	0	12	8	0	0	9	7
π_1:2	0	0	0	0	0	0	0	0
π_2:2	0	0	0	0	0	0	0	0
π_1:1	1	1	3	1	1	1	2	1
π_2:1	1	1	3	1	1	1	2	1

Munkres Chapter 16

	Nesting				Alternating Nesting			
Definiendum	LPT	α	$\mathscr{E}(\alpha, \varnothing)$	$\mathscr{E}(\alpha, B_0)$	LPT	α	$\mathscr{E}(\alpha, \varnothing)$	$\mathscr{E}(\alpha, B_0)$
`Subspacetop`:3	0	2	4	2	0	2	4	2
`SUBSPACE`:4	0	0	4	2	0	0	4	2
`Dictionaryorder`:4	2	3	7	3	2	3	4	3
`Orderedsquare`:0	0	2	196	96	0	2	90	56
`CONVEX`:3	1	1	6	3	1	1	4	2

Munkres Chapter 17

	Nesting				Alternating Nesting			
Definiendum	LPT	α	$\mathscr{E}(\alpha, \varnothing)$	$\mathscr{E}(\alpha, B_0)$	LPT	α	$\mathscr{E}(\alpha, \varnothing)$	$\mathscr{E}(\alpha, B_0)$
CLOSEDSET:3	0	0	6	1	0	0	5	1
Interior:1	0	2	6	2	0	2	5	2
Closure:3	0	2	11	3	0	2	9	3
INTERSECTS:2	0	0	2	0	0	0	2	0
DISJOINT:2	0	0	2	0	0	0	2	0
NBHD:4	0	0	4	1	0	0	3	1
LIMITPT:4	1	2	6	2	1	1	5	2
TOPCONV:3	3	3	24	14	3	3	18	13
HAUSDORFF:2	2	2	4	2	2	2	4	2
TOPLIMIT:3	1	2	26	16	1	1	19	14

Munkres Chapter 18

	Nesting				Alternating Nesting			
Definiendum	LPT	α	$\mathscr{E}(\alpha, \varnothing)$	$\mathscr{E}(\alpha, B_0)$	LPT	α	$\mathscr{E}(\alpha, \varnothing)$	$\mathscr{E}(\alpha, B_0)$
CONTINUOUS:5	1	1	21	11	1	1	14	8
CONTAT:6	2	2	19	10	2	2	13	8
HOMEOMORPHISM:5	0	0	25	13	0	0	17	10
HOMEOMORPHIC:4	1	1	26	14	1	1	18	11
TOPIMBED:5	0	0	34	17	0	0	24	13

Munkres Chapter 19

	Nesting				Alternating Nesting			
Definiendum	LPT	α	$\mathscr{E}(\alpha, \varnothing)$	$\mathscr{E}(\alpha, B_0)$	LPT	α	$\mathscr{E}(\alpha, \varnothing)$	$\mathscr{E}(\alpha, B_0)$
TUPLE:3	1	1	5	3	1	1	3	2
Cartesprod:2	0	2	7	5	0	2	5	4
Cartespow:2	0	2	11	7	0	2	8	6
Boxtopbasis:3	1	3	9	7	1	3	7	6
Boxtop:3	1	1	17	11	1	1	12	9
Projection:3	0	2	9	7	0	2	7	6
Prodtopsubbasis:3	1	2	31	19	1	2	23	16
Productspace:3	1	1	47	29	1	1	33	24

Munkres Chapter 20

	Nesting				Alternating Nesting			
Definiendum	LPT	α	$\mathscr{E}(\alpha, \varnothing)$	$\mathscr{E}(\alpha, B_0)$	LPT	α	$\mathscr{E}(\alpha, \varnothing)$	$\mathscr{E}(\alpha, B_0)$
METRIC:2	1	1	328	151	1	1	127	75
Ball:4	0	2	328	151	0	2	127	75
Metrictopbasis:2	0	2	330	153	0	2	129	77
Metrictop:2	0	0	336	157	0	0	133	80
METRIZABLE:2	1	1	337	158	1	1	134	80
METRICSPACE:3	0	0	336	157	0	0	133	80
BOUNDED:3	2	2	336	157	2	2	133	80
Diam:1	0	3	336	157	0	3	133	80
Stdbddmetric:2	1	3	336	157	1	2	133	80
Rnnorm:2	0	2	704	318	0	2	242	139
Rneuclideanmetric:3	0	0	912	409	0	0	314	178
Rnsqmetric:3	0	2	546	249	0	2	197	114
Uniformmetric:3	0	2	1235	552	0	2	422	239

Munkres Chapter 21

	Nesting				Alternating Nesting			
Definiendum	LPT	α	$\mathscr{E}(\alpha, \varnothing)$	$\mathscr{E}(\alpha, B_0)$	LPT	α	$\mathscr{E}(\alpha, \varnothing)$	$\mathscr{E}(\alpha, B_0)$
CNTBLBASISAT:3	3	3	23	13	3	3	17	12
FIRSTCNTBL:2	1	1	24	14	1	1	18	13
UNIFCONV:7	4	4	336	157	3	3	133	80

Munkres Chapter 22

	Nesting				Alternating Nesting			
Definiendum	LPT	α	$\mathscr{E}(\alpha, \varnothing)$	$\mathscr{E}(\alpha, B_0)$	LPT	α	$\mathscr{E}(\alpha, \varnothing)$	$\mathscr{E}(\alpha, B_0)$
QUOTIENTMAP:5	1	1	21	11	1	1	14	8
SATURATED:5	1	2	23	11	1	1	16	9
OPENMAP:5	1	1	17	9	1	1	11	6
CLOSEDMAP:5	1	1	23	10	1	1	17	7
Quotienttop:3	0	1	22	12	0	1	15	9
QUOTIENTSPACE:2	1	2	22	12	1	2	15	9

Munkres Chapter 23

Definiendum	**Nesting** LPT	α	$\mathscr{E}(\alpha, \varnothing)$	$\mathscr{E}(\alpha, B_0)$	**Alternating Nesting** LPT	α	$\mathscr{E}(\alpha, \varnothing)$	$\mathscr{E}(\alpha, B_0)$
SEPARATION:4	0	0	4	1	0	0	3	1
CONNECTED:2	1	1	5	2	1	1	4	2
TOTALLYDISCONNECTED:2	1	1	13	7	1	1	11	7

Munkres Chapter 24

Definiendum	**Nesting** LPT	α	$\mathscr{E}(\alpha, \varnothing)$	$\mathscr{E}(\alpha, B_0)$	**Alternating Nesting** LPT	α	$\mathscr{E}(\alpha, \varnothing)$	$\mathscr{E}(\alpha, B_0)$
LINEARCONTINUUM:2	2	2	13	7	2	2	10	6
PATH:5	1	3	224	114	1	3	111	68
PATHCONNECTED:2	2	2	226	116	2	2	112	69
Rnunitball:1	0	2	802	362	0	2	274	157
Puncturedeuclideanspace:1	0	1	215	109	0	1	104	65
Unitsphere:1	0	2	706	320	0	2	243	140

Munkres Chapter 25

Definiendum	**Nesting** LPT	α	$\mathscr{E}(\alpha, \varnothing)$	$\mathscr{E}(\alpha, B_0)$	**Alternating Nesting** LPT	α	$\mathscr{E}(\alpha, \varnothing)$	$\mathscr{E}(\alpha, B_0)$
Ccequiv:2	1	3	20	14	1	2	16	12
$\sim_{\mathtt{CC}}$:infix	0	0	20	14	0	0	16	12
Components:2	0	0	33	20	0	0	26	17
Pcequiv:2	1	3	227	117	1	2	112	69
$\sim_{\mathtt{PC}}$:infix	0	0	227	117	0	0	112	69
Pathcomponents:2	0	0	240	123	0	0	122	74
LOCALCONNAT:3	2	2	11	6	2	2	9	5
LOCALCONN:2	1	1	12	7	1	1	9	5
LOCALPATHCONNAT:3	2	2	228	118	2	2	114	71
LOCALPATHCONN:2	1	1	229	119	1	1	114	71
WEAKLOCALCONNAT:3	3	3	11	6	2	2	9	5

Munkres Chapter 26

	Nesting				Alternating Nesting			
Definiendum	LPT	α	$\mathscr{E}(\alpha, \varnothing)$	$\mathscr{E}(\alpha, B_0)$	LPT	α	$\mathscr{E}(\alpha, \varnothing)$	$\mathscr{E}(\alpha, B_0)$
COVER:2	0	0	4	0	0	0	3	0
OPENCOVER:3	0	0	4	1	0	0	3	1
COMPACT:2	2	2	12	8	2	2	10	8
COVER:3	0	0	4	0	0	0	3	0
FINITEINTERSPROP:1	1	1	11	7	1	1	8	6

Munkres Chapter 27

	Nesting				Alternating Nesting			
Definiendum	LPT	α	$\mathscr{E}(\alpha, \varnothing)$	$\mathscr{E}(\alpha, B_0)$	LPT	α	$\mathscr{E}(\alpha, \varnothing)$	$\mathscr{E}(\alpha, B_0)$
Distance:4	0	2	336	157	0	2	133	80
LEBESGUENUM:4	2	2	433	200	2	2	165	98
UNIFORMCONT:5	3	3	336	157	3	3	133	80
ISOLATEDPT:3	0	1	4	1	0	1	3	1

Munkres Chapter 28

	Nesting				Alternating Nesting			
Definiendum	LPT	α	$\mathscr{E}(\alpha, \varnothing)$	$\mathscr{E}(\alpha, B_0)$	LPT	α	$\mathscr{E}(\alpha, \varnothing)$	$\mathscr{E}(\alpha, B_0)$
LIMITPTCPT:2	2	2	8	4	2	2	7	4
SUBSEQ:2	1	1	24	14	1	1	18	13
SEQCPT:2	3	3	27	17	2	2	20	15
CNTBLCPT:2	2	2	26	15	2	2	21	15

Munkres Chapter 29

	Nesting				Alternating Nesting			
Definiendum	LPT	α	$\mathscr{E}(\alpha, \varnothing)$	$\mathscr{E}(\alpha, B_0)$	LPT	α	$\mathscr{E}(\alpha, \varnothing)$	$\mathscr{E}(\alpha, B_0)$
LOCCPTAT:3	1	1	17	11	1	1	15	11
LOCCPT:2	1	1	18	12	1	1	16	12
COMPACTIFICATION:4	0	0	12	8	0	0	10	8
Oneptcompactification:2	0	1	13	9	0	1	11	9

Munkres Chapter 30

	Nesting				Alternating Nesting			
Definiendum	LPT	α	$\mathscr{E}(\alpha, \varnothing)$	$\mathscr{E}(\alpha, B_0)$	LPT	α	$\mathscr{E}(\alpha, \varnothing)$	$\mathscr{E}(\alpha, B_0)$
SECONDCNTBL:2	1	1	26	15	1	1	20	14
DENSE:3	0	0	11	3	0	0	9	3
LINDELOF:2	2	2	27	16	2	2	21	15
SEPARABLESPACE:2	1	1	26	15	1	1	20	14
Sorgenfreyplane:0	0	0	204	102	0	0	95	59
GDELTA:3	1	3	26	15	1	3	20	14

Munkres Chapter 31

	Nesting				Alternating Nesting			
Definiendum	LPT	α	$\mathscr{E}(\alpha, \varnothing)$	$\mathscr{E}(\alpha, B_0)$	LPT	α	$\mathscr{E}(\alpha, \varnothing)$	$\mathscr{E}(\alpha, B_0)$
REGULARSP:2	3	3	8	3	2	2	7	2
NORMALSP:2	2	2	8	3	2	2	7	2

Munkres Chapter 32

	Nesting				Alternating Nesting			
Definiendum	LPT	α	$\mathscr{E}(\alpha, \varnothing)$	$\mathscr{E}(\alpha, B_0)$	LPT	α	$\mathscr{E}(\alpha, \varnothing)$	$\mathscr{E}(\alpha, B_0)$
COMPLETELYNORMALSP:2	1	1	12	5	1	1	10	4

Munkres Chapter 33

	Nesting				Alternating Nesting			
Definiendum	LPT	α	$\mathscr{E}(\alpha, \varnothing)$	$\mathscr{E}(\alpha, B_0)$	LPT	α	$\mathscr{E}(\alpha, \varnothing)$	$\mathscr{E}(\alpha, B_0)$
SEPBYCONTFUNC:4	1	3	224	114	1	3	111	68
COMPLETELYREGULARSP:2	2	3	226	116	1	1	112	69
PERFECTLYNORMALSP:2	1	1	27	16	1	1	21	15

Munkres Chapter 34

	Nesting				Alternating Nesting			
Definiendum	LPT	α	$\mathscr{E}(\alpha, \varnothing)$	$\mathscr{E}(\alpha, B_0)$	LPT	α	$\mathscr{E}(\alpha, \varnothing)$	$\mathscr{E}(\alpha, B_0)$
LOCALLYMETRIZABLE:2	2	2	339	160	2	2	135	81

Munkres Chapter 35

Definiendum	Nesting				Alternating Nesting			
	LPT	α	$\mathscr{E}(\alpha, \varnothing)$	$\mathscr{E}(\alpha, B_0)$	LPT	α	$\mathscr{E}(\alpha, \varnothing)$	$\mathscr{E}(\alpha, B_0)$
EXTENDS:infix	1	1	8	5	1	1	5	4
UNIVEXTPROP:2	2	2	26	14	2	2	18	10

Munkres Chapter 36

Definiendum	Nesting				Alternating Nesting			
	LPT	α	$\mathscr{E}(\alpha, \varnothing)$	$\mathscr{E}(\alpha, B_0)$	LPT	α	$\mathscr{E}(\alpha, \varnothing)$	$\mathscr{E}(\alpha, B_0)$
Manifold:1	3	5	244	128	2	4	125	79
TOPCURVE:2	0	0	244	128	0	0	125	79
TOPSURFACE:2	0	0	244	128	0	0	125	79
Support:1	0	1	217	103	0	1	108	62
PARTITIONOFUNITY:4	1	3	235	112	1	2	98	60
POUDOMBY:5	1	2	235	112	1	2	109	62
POINTFINFAM:3	1	3	15	9	1	2	11	8

Munkres Chapter 37

Definiendum	Nesting				Alternating Nesting			
	LPT	α	$\mathscr{E}(\alpha, \varnothing)$	$\mathscr{E}(\alpha, B_0)$	LPT	α	$\mathscr{E}(\alpha, \varnothing)$	$\mathscr{E}(\alpha, B_0)$
CNTBLINTERSPROP:1	1	1	26	15	1	1	21	15

Munkres Chapter 38

Definiendum	Nesting				Alternating Nesting			
	LPT	α	$\mathscr{E}(\alpha, \varnothing)$	$\mathscr{E}(\alpha, B_0)$	LPT	α	$\mathscr{E}(\alpha, \varnothing)$	$\mathscr{E}(\alpha, B_0)$
EQUIVCOMPACTIFICATION:6	2	2	26	14	2	2	18	11
Imbedinducedcptfcation:1	1	3	37	20	1	2	26	15
Stonecech:2	2	5	226	116	2	5	112	69

Appendix E

Symbol count data

We display three counts for each definition α: (1) In the column headed **0** we give the number of symbols in the definiens of α; (2) in the column headed **A** we give the number of symbols in $\mathscr{E}(\alpha, \varnothing)$, i.e., the *full* expansion of α; (3) in the column headed **B** we give the number of symbols in $\mathscr{E}(\alpha, B_0)$, where B_0 is the set of foundational definitions given in Section 4.2.

In these tables, we put a ceiling on the symbol count of any expanded definition: at $2^{31}-1$ symbols (the maximum value for an integer in the Java programming language), we say "enough is enough". In the tables, this is recorded as '2147483647+'. We could have easily gone to higher numbers, but the point is already made: the raw symbol count for expanded definitions goes exponentially, and rapidly gets out of hand.

E.1 Foundations – Suppes

Suppes Chapter 2

Definiendum	**0**	**A**	**B**
Δ_0:infix	31	300	31
$\times$:infix	112	270	112

Suppes Chapter 3

Definiendum	**0**	**A**	**B**
BR:1	66	224	66
TR:1	83	571	83
Dom:1	141	518	202
Rng:1	141	518	202
Fld:1	19	1128	411
Cnv:1	146	681	207
$\circ$:infix	199	1111	321
$\mid$:infix	61	1138	419
$\mid_{\mathtt{Rng}}$:infix	46	3046	1134
RFX:2	67	444	128
IRFX:2	73	450	134
SYM:2	123	658	184
ASYM:2	125	660	186
ANSYM:2	132	667	193
TRAN:2	166	859	227
CONN:2	136	671	197
SCONN:2	112	647	173
RFX:1	26	1803	612
IRFX:1	27	1809	618
SYM:1	26	1924	660
ASYM:1	27	1926	662
ANSYM:1	28	1933	669
TRAN:1	27	2642	904
CONN:1	27	1937	673
SCONN:1	28	1913	649
Id:1	95	253	95
QORD:2	25	1311	363
PORD:2	43	1986	564

Suppes Chapter 3, continued

Definiendum	0	A	B
SORD:2	45	2189	609
SPORD:2	26	1527	421
SSORD:2	43	2206	626
QORD:1	14	5819	2003
PORD:1	14	8748	3024
SORD:1	14	10078	3479
SPORD:1	15	7162	2471
SSORD:1	15	10095	3496
MELT:3	88	465	149
FELT:3	106	483	167
WO:2	113	1303	439
ISUC:3	119	812	180
LELT:3	106	483	167
SECT:3	73	9576	3105
Seg:3	147	524	208
LB:3	67	444	128
INF:3	93	1121	331
UB:3	67	444	128
SUP:3	93	1121	331
EQUIV:1	35	6385	2192
EQUIV:2	26	7525	2615
Coset:2	153	7810	2742
PART:2	179	468	179
PART:1	20	479	190
FINER:2	126	1094	492
Part:1	133	15432	5454
Reln:1	175	805	358
FCN:1	130	446	130
MONO:1	25	2260	680
FCN:3	51	1539	567
SURJ:3	42	1506	558
MONO:3	52	3353	1117
BIJ:3	43	3320	1108
Maps:2	84	1613	641

Suppes Chapter 4

Definiendum	0	A	B
$\approx_{\mathtt{C}}$:infix	21	3331	1119
$\leq_{\mathtt{C}}$:infix	51	3390	1154
$<_{\mathtt{C}}$:infix	40	6794	2322
MNEL:2	61	61	61
MXEL:2	61	61	61
FIN:1	116	348	168
DFIN:1	86	3425	1189

Suppes Chapter 5

Definiendum	0	A	B
TRANS:1	73	73	73
ECONN:1	82	82	82
ORD:1	24	163	163
Eps:1	127	285	127
$<_{\mathtt{O}}$:infix	35	349	349
$\leq_{\mathtt{O}}$:infix	46	360	360
$>_{\mathtt{O}}$:infix	35	349	349
$\geq_{\mathtt{O}}$:infix	46	360	360
Suc:1	123	627	627
NAT:1	31	7714	2900
ω:0	80	7788	2974
$\mathbb{N}$:0	6	7788	2974
$0_{\mathtt{N}}$:0	11	61	11
$1_{\mathtt{N}}$:0	61	111	61
$2_{\mathtt{N}}$:0	10	1727	1227
$3_{\mathtt{N}}$:0	10	17887	12887
$4_{\mathtt{N}}$:0	10	179487	129487
$5_{\mathtt{N}}$:0	10	1795487	1295487
$6_{\mathtt{N}}$:0	10	17955487	12955487
$7_{\mathtt{N}}$:0	10	179555487	129555487
$8_{\mathtt{N}}$:0	10	1795555487	1295555487
$9_{\mathtt{N}}$:0	10	2147483647+	2147483647+
$10_{\mathtt{N}}$:0	10	2147483647+	2147483647+

Suppes Chapter 5, continued

Definiendum	0	A	B
Addnat:0	402	40263	13927
$+_{N}$:infix	77	40822	13998
Mulnat:0	418	158150	40334
$\cdot_{N}$:infix	77	158709	40405
Expnat:0	422	276087	66791
Exp_{N}:infix	77	276646	66862
INF:1	12	354	174
DEN:1	21	11118	4092
DINF:1	13	3431	1195
CNTBL:1	26	11118	4092
UNCNTBL:1	14	11124	4098

Suppes Chapter 6

Definiendum	0	A	B
/:infix	100	15878	6042
Fr:0	107	31851	12179
$\equiv_{\text{Fr}}$:infix	87	349221	92941
$<_{\text{Fr}}$:infix	92	1619229	416517
$>_{\text{Fr}}$:infix	11	1619229	416517
$\leq_{\text{Fr}}$:infix	33	1968456	509464
$\geq_{\text{Fr}}$:infix	33	1968456	509464
$+_{\text{Fr}}$:infix	183	3067779	153452
$\cdot_{\text{Fr}}$:infix	161	365167	99051
Nra:0	196	10873248	2898930
$\text{Incl}_{\text{FrNra}}$:1	153	10873205	2898887
$<_{\text{Nra}}$:infix	89	23365797	6214449
$>_{\text{Nra}}$:infix	89	23365797	6214449
$\leq_{\text{Nra}}$:infix	92	23715024	6307396
$\geq_{\text{Nra}}$:infix	92	23715024	6307396
$+_{\text{Nra}}$:infix	176	36036884	8943323
$\cdot_{\text{Nra}}$:infix	180	33334272	8888922
0_{Nra}:0	128	10890479	2899154
1_{Nra}:0	128	10890929	2899354
$\equiv_{\text{SUB}}$:infix	101	72074161	17886723

Suppes Chapter 6, continued

Definiendum	0	A	B
$<_{\texttt{SUB}}$:infix	108	167513719	41987811
$+_{\texttt{SUB}}$:infix	193	216221921	53660081
$\cdot_{\texttt{SUB}}$:infix	261	749570289	195882849
Ra:0	212	2147483647+	560291965
$\texttt{Incl}_{\texttt{NraRa}}$:1	173	2147483647+	563191212
$\texttt{Incl}_{\texttt{FrRa}}$:1	65	2147483647+	571900083
$\mathbb{Q}$:0	2	2147483647+	560291965
$<_{\texttt{Ra}}$:infix	88	2147483647+	1162571813
$+_{\texttt{Ra}}$:infix	163	2147483647+	1734536121
$\cdot_{\texttt{Ra}}$:infix	167	2147483647+	1876758889
$0_{\texttt{Ra}}$:0	142	2147483647+	566090554
$1_{\texttt{Ra}}$:0	142	2147483647+	566090954
$>_{\texttt{Ra}}$:infix	11	2147483647+	1162571813
$\leq_{\texttt{Ra}}$:infix	22	2147483647+	1162571824
$\geq_{\texttt{Ra}}$:infix	22	2147483647+	1162571824
$-_{\texttt{Ra}}$:infix	52	2147483647+	1734536162
$\texttt{Av}_{\texttt{Ra}}$:1	149	2147483647+	2147483647+
$\texttt{Nat}_{\texttt{Ra}}$:0	134	2147483647+	2147483647+
$\mathbb{N}_{\texttt{Ra}}$:0	8	2147483647+	2147483647+
$\texttt{Int}_{\texttt{Ra}}$:0	121	2147483647+	2147483647+
$\mathbb{Z}_{\texttt{Ra}}$:0	8	2147483647+	2147483647+
$\texttt{Seq}_{\texttt{Ra}}$:0	15	2147483647+	560295578
$+_{\texttt{SeqRa}}$:infix	249	2147483647+	2147483647+
$\cdot_{\texttt{SeqRa}}$:infix	253	2147483647+	2147483647+
$<_{\texttt{N}}$:infix	50	15953	6325
$>_{\texttt{N}}$:infix	50	15953	6325
$\leq_{\texttt{N}}$:infix	21	15964	6336
$\geq_{\texttt{N}}$:infix	21	15964	6336
$\texttt{Cseq}_{\texttt{Ra}}$:0	322	2147483647+	2147483647+
$\equiv_{\texttt{CseqRa}}$:infix	203	2147483647+	2147483647+
$<_{\texttt{CseqRa}}$:infix	235	2147483647+	2147483647+
$\mathbb{R}$:0	207	2147483647+	2147483647+
$<_{\texttt{R}}$:infix	107	2147483647+	2147483647+
$>_{\texttt{R}}$:infix	10	2147483647+	2147483647+
$\leq_{\texttt{R}}$:infix	21	2147483647+	2147483647+
$\geq_{\texttt{R}}$:infix	21	2147483647+	2147483647+
$+_{\texttt{R}}$:infix	204	2147483647+	2147483647+
$-_{\texttt{R}}$:infix	51	2147483647+	2147483647+
$\cdot_{\texttt{R}}$:infix	208	2147483647+	2147483647+

Suppes Chapter 6, continued

Definiendum	0	A	B
0_R:0	170	2147483647+	2147483647+
1_R:0	170	2147483647+	2147483647+
Av_R:1	151	2147483647+	2147483647+
Id_R:1	165	2147483647+	2147483647+
$Incl_{NraR}$:1	60	2147483647+	2147483647+
$Incl_{FrR}$:1	58	2147483647+	2147483647+
Ra_R:0	90	2147483647+	2147483647+
Seq_R:0	23	2147483647+	2147483647+
$Cseq_R$:0	318	2147483647+	2147483647+
Lim:1	271	2147483647+	2147483647+
UB_R:2	101	2147483647+	2147483647+
Min_R:1	154	2147483647+	2147483647+
Max_R:1	154	2147483647+	2147483647+
Lub_R:1	145	2147483647+	2147483647+
$Finitesumset_R$:1	526	2147483647+	2147483647+
$Finitesum_R$:1	187	2147483647+	2147483647+
$Sqrt_R$:1	150	2147483647+	2147483647+
Sup_R:1	146	2147483647+	2147483647+
Inf_R:1	146	2147483647+	2147483647+

E.2 Topology – Munkres

Munkres Chapter 12

Definiendum	0	A	B
TOPOLOGY:2	194	587	194
TOPSP:2	13	587	194
OPENSET:3	25	602	209
FINERTOP:3	52	1230	420
STRFINERTOP:3	71	1249	439
COARSERTOP:3	52	1230	420
STRCOARSERTOP:3	71	1249	439

Munkres Chapter 13

Definiendum	0	A	B
TOPBASIS:2	252	493	252
Basisgentop:2	261	919	500
Stdrealtopbasis:0	303	2147483647+	2147483647+
Stdrealtop:0	39	2147483647+	2147483647+
Lowerlimitrealtop:0	330	2147483647+	2147483647+
Krealtop:0	319	2147483647+	2147483647+
TOPSUBBASIS:2	38	270	38
Subbasisgentop:2	320	4101	1502

Munkres Chapter 14

Definiendum	0	A	B
Oointerval:4	209	2721	825
OOINTERVAL:3	111	5033	1533
Ocinterval:4	220	2732	836
OCINTERVAL:3	111	5044	1544
Cointerval:4	220	2732	836
COINTERVAL:3	111	5044	1544
Ccinterval:4	231	2743	847
CCINTERVAL:3	111	5055	1555
Ordertopbasis:2	277	11545	3645
Ordertop:2	71	72417	23018
Oplusray:3	165	2519	781
Ominusray:3	165	2519	781
Cplusray:3	176	2530	792
Cminusray:3	176	2530	792

Munkres Chapter 15

Definiendum	0	A	B
Prodtopbasis:4	164	1577	633
Prodtop:4	92	11585	4716
π_1:2	1	1	1
π_2:2	1	1	1
π_1:1	64	222	64
π_2:1	64	222	64

Munkres Chapter 16

Definiendum	0	A	B
Subspacetop:3	151	839	335
SUBSPACE:4	64	1486	565
Dictionaryorder:4	324	5678	1556
Orderedsquare:0	448	2147483647+	2147483647+
CONVEX:3	125	5071	1547

Munkres Chapter 17

Definiendum	0	A	B
CLOSEDSET:3	65	1346	444
Interior:1	171	898	355
Closure:3	178	2240	790
INTERSECTS:2	29	166	29
DISJOINT:2	23	160	23
NBHD:4	57	634	241
LIMITPT:4	168	1633	594
TOPCONV:3	237	34696	13524
HAUSDORFF:2	184	908	378
TOPLIMIT:3	112	70356	27482

Munkres Chapter 18

Definiendum	0	A	B
CONTINUOUS:5	117	12024	4088
CONTAT:6	201	7606	2810
HOMEOMORPHISM:5	111	41496	13620
HOMEOMORPHIC:4	73	42699	14037
TOPIMBED:5	137	97964	34966

Munkres Chapter 19

Definiendum	0	A	B
TUPLE:3	168	2388	808
Cartesprod:2	138	3466	1254
Cartespow:2	125	5878	2244
Boxtopbasis:3	421	9449	3052
Boxtop:3	272	86540	27607
Projection:3	223	4942	1782
Prodtopsubbasis:3	362	102666	36231
Productspace:3	279	121838	41520

Munkres Chapter 20

Definiendum	0	A	B
METRIC:2	627	2147483647+	2147483647+
Ball:4	222	2147483647+	2147483647+
Metrictopbasis:2	160	2147483647+	2147483647+
Metrictop:2	73	2147483647+	2147483647+
METRIZABLE:2	49	2147483647+	2147483647+
METRICSPACE:3	38	2147483647+	2147483647+
BOUNDED:3	196	2147483647+	2147483647+
Diam:1	409	2147483647+	2147483647+
Stdbddmetric:2	340	2147483647+	2147483647+
Rnnorm:2	288	2147483647+	2147483647+
Rneuclideanmetric:3	134	2147483647+	2147483647+
Rnsqmetric:3	287	2147483647+	2147483647+
Uniformmetric:3	474	2147483647+	2147483647+

Munkres Chapter 21

Definiendum	0	A	B
CNTBLBASISAT:3	196	18661	7093
FIRSTCNTBL:2	72	19291	7330
UNIFCONV:7	501	2147483647+	2147483647+

Munkres Chapter 22

Definiendum	0	A	B
QUOTIENTMAP:5	144	11966	4030
SATURATED:5	286	21532	7257
OPENMAP:5	107	5824	2154
CLOSEDMAP:5	125	11547	4161
Quotienttop:3	137	14727	5057
QUOTIENTSPACE:2	227	17608	6150

Munkres Chapter 23

Definiendum	0	A	B
SEPARATION:4	142	1051	336
CONNECTED:2	59	1668	560
TOTALLYDISCONNECTED:2	124	20867	8008

Munkres Chapter 24

Definiendum	0	A	B
LINEARCONTINUUM:2	370	11660	3854
PATH:5	510	2147483647+	2147483647+
PATHCONNECTED:2	97	2147483647+	2147483647+
Rnunitball:1	184	2147483647+	2147483647+
Puncturedeuclideanspace:1	290	2147483647+	2147483647+
Unitsphere:1	198	2147483647+	2147483647+

Munkres Chapter 25

Definiendum	0	A	B
Ccequiv:2	292	48386	25947
$\sim_{\mathtt{CC}}$:infix	82	49191	26201
Components:2	56	1274057	680272
Pcequiv:2	202	2147483647+	2147483647+
$\sim_{\mathtt{PC}}$:infix	82	2147483647+	2147483647+
Pathcomponents:2	56	2147483647+	2147483647+
LOCALCONNAT:3	161	18725	7342
LOCALCONN:2	71	19355	7579
LOCALPATHCONNAT:3	165	2147483647+	2147483647+
LOCALPATHCONN:2	75	2147483647+	2147483647+
WEAKLOCALCONNAT:3	196	18784	7377

Munkres Chapter 26

Definiendum	0	A	B
COVER:2	40	272	40
OPENCOVER:3	52	915	266
COMPACT:2	128	5945	1994
COVER:3	71	351	71
FINITEINTERSPROP:1	87	3686	1269

Munkres Chapter 27

Definiendum	0	A	B
Distance:4	251	2147483647+	2147483647+
LEBESGUENUM:4	205	2147483647+	2147483647+
UNIFORMCONT:5	388	2147483647+	2147483647+
ISOLATEDPT:3	92	669	276

Munkres Chapter 28

Definiendum	0	A	B
LIMITPTCPT:2	123	2689	1053
SUBSEQ:2	241	45966	17582
SEQCPT:2	140	90719	34984
CNTBLCPT:2	149	17076	6099

Munkres Chapter 29

Definiendum	0	A	B
LOCCPTAT:3	167	29141	11228
LOCCPT:2	68	29771	11465
COMPACTIFICATION:4	118	10640	3788
Oneptcompactification:2	153	15016	5259

Munkres Chapter 30

Definiendum	0	A	B
SECONDCNTBL:2	67	12657	4819
DENSE:3	60	2887	1020
LINDELOF:2	129	13638	4897
SEPARABLESPACE:2	83	14669	5359
Sorgenfreyplane:0	85	2147483647+	2147483647+
GDELTA:3	219	13799	5119

Munkres Chapter 31

Definiendum	0	A	B
REGULARSP:2	346	3828	1446
NORMALSP:2	369	5376	1907

Munkres Chapter 32

Definiendum	0	A	B
COMPLETELYNORMALSP:2	90	57155	22525

Munkres Chapter 33

Definiendum	0	A	B
SEPBYCONTFUNC:4	714	2147483647+	2147483647+
COMPLETELYREGULARSP:2	322	2147483647+	2147483647+
PERFECTLYNORMALSP:2	96	21152	7708

Munkres Chapter 34

Definiendum	0	A	B
LOCALLYMETRIZABLE:2	119	2147483647+	2147483647+

Munkres Chapter 35

Definiendum	0	A	B
EXTENDS:infix	170	3718	1434
UNIVEXTPROP:2	188	42688	15232

Munkres Chapter 36

Definiendum	0	A	B
Manifold:1	396	2147483647+	2147483647+
TOPCURVE:2	35	2147483647+	2147483647+
TOPSURFACE:2	35	2147483647+	2147483647+
Support:1	174	2147483647+	2147483647+
PARTITIONOFUNITY:4	592	2147483647+	2147483647+
POUDOMBY:5	313	2147483647+	2147483647+
POINTFINFAM:3	198	6056	2268

Munkres Chapter 37

Definiendum	0	A	B
CNTBLINTERSPROP:1	88	11379	4172

Munkres Chapter 38

Definiendum	0	A	B
EQUIVCOMPACTIFICATION:6	182	63042	21304
Imbedinducedcptfcation:1	222	210584	75294
Stonecech:2	378	2147483647+	2147483647+

Appendix F

Definitions: Foundations – Suppes

In his paper on the Proofless Text system [11], Friedman formalizes the definitions from Chapters 2 through 6 of Suppes's *Axiomatic Set Theory*, [15]. I have parsed and translated all of those definitions, with a few additional ones added in, as they were needed for the Munkres Topology definitions.

For each of these definitions, find below a triplet, consisting of:

i. A transcription of the definition into LPT input,

ii. The LaTeX version of the input, and

iii. The DZFC translation of the definition, as performed by `lpt2dzfc`.

Side-by-side comparison of these definitions against the originals as they appear in [15] will reveal how good of an approximation to the original is possible in LPT.

F.1 Chapter 2

```
DEFINITION FS.1.1: Infix function \Delta_{0}. x \Delta_{0} y \simeq
(x \less y) \cup (y \less x). Precedence 60.
```

DEFINITION FS.1.1: Infix function Δ_0. $x \,\Delta_0\, y \simeq (x \backslash y) \cup (y \backslash x)$. Precedence 60.

$\Delta_0(x, y) \simeq \cup((\backslash(x, y)), (\backslash(y, x)))$

```
DEFINITION FS.1.2: Infix function \times. x \times y \simeq {
  <z,w> : z \in x \wedge w \in y
}. Precedence 20.
```

DEFINITION FS.1.2: Infix function $\times$. $x \times y \simeq \{\langle z, w\rangle : z \in x \wedge w \in y\}$. Precedence 20.

$\times(x, y) \simeq (\iota y_0)(\forall x_0)(x_0 \in y_0 \leftrightarrow (\exists z, w)(x_0 = \varpi_0(z, w) \wedge (z \in x \wedge w \in y)))$

F.2 Chapter 3

```
DEFINITION FS.2.1: 1-ary relation BR. BR[A] \iff (\forall y \in A)
(\exists z,w)(y = <z,w>).
```

DEFINITION FS.2.1: 1-ary relation BR. $\mathtt{BR}[A] \leftrightarrow (\forall y \in A)(\exists z, w)(y = \langle z, w\rangle)$.

$\mathtt{BR}[A] \leftrightarrow (\forall y)(y \in A \rightarrow (\exists z, w)y = \varpi_0(z, w))$

```
DEFINITION FS.2.2: 1-ary relation TR. TR[A] \iff (\forall y \in A)
(\exists z,w,u)(y = <z,w,u>).
```

DEFINITION FS.2.2: 1-ary relation TR. $\mathtt{TR}[A] \leftrightarrow (\forall y \in A)(\exists z, w, u)(y = \langle z, w, u\rangle)$.

$\mathtt{TR}[A] \leftrightarrow (\forall y)(y \in A \rightarrow (\exists z, w, u)y = \varpi_0(\varpi_0(z, w), u))$

```
DEFINITION FS.2.3: 1-ary function Dom. If BR[R] then Dom(R) \simeq
{x : (\exists y)(x \infixrl{R} y)}. Otherwise Dom(R) \up.
```

DEFINITION FS.2.3: 1-ary function Dom. If $\mathtt{BR}[R]$ then $\mathtt{Dom}(R) \simeq \{x : (\exists y)(xRy)\}$. Otherwise $\mathtt{Dom}(R)\uparrow$.

$\mathtt{Dom}(R) \simeq (\iota z_0)((\mathtt{BR}[R] \wedge z_0 \simeq (\iota y_0)(\forall x_0)(x_0 \in y_0 \leftrightarrow (\exists x)(x_0 = x \wedge ((\exists y)\, \varpi_0(x, y) \in R)))))$

```
DEFINITION FS.2.4: 1-ary function Rng. If BR[R] then Rng(R) \simeq
{y : (\exists x)(x \infixrl{R} y)}. Otherwise Rng(R) \up.
```

DEFINITION FS.2.4: 1-ary function `Rng`. If `BR`$[R]$ then `Rng`$(R) \simeq \{y : (\exists x)(xRy)\}$. Otherwise `Rng`$(R)\uparrow$.

`Rng`$(R) \simeq (\iota z_0)((\texttt{BR}[R] \wedge z_0 \simeq (\iota y_0)(\forall x_0)(x_0 \in y_0 \leftrightarrow (\exists y)(x_0 = y \wedge ((\exists x)\, \varpi_0(x, y) \in R)))))$

```
DEFINITION FS.2.5: 1-ary function Fld. Fld(R) \simeq
Dom(R) \cup Rng(R).
```

DEFINITION FS.2.5: 1-ary function `Fld`. `Fld`$(R) \simeq$ `Dom`$(R) \cup$ `Rng`(R).

`Fld`$(R) \simeq \cup($`Dom`$(R),$ `Rng`$(R))$

```
DEFINITION FS.2.6: 1-ary function Cnv. If BR[R] then Cnv(R) \simeq
{<x,y> : y \infixrl{R} x}. Otherwise Cnv(R) \up.
```

DEFINITION FS.2.6: 1-ary function `Cnv`. If `BR`$[R]$ then `Cnv`$(R) \simeq \{\langle x, y\rangle : yRx\}$. Otherwise `Cnv`$(R)\uparrow$.

`Cnv`$(R) \simeq (\iota z_0)((\texttt{BR}[R] \wedge z_0 \simeq (\iota y_0)(\forall x_0)(x_0 \in y_0 \leftrightarrow (\exists x, y)(x_0 = \varpi_0(x, y) \wedge (\varpi_0(y, x) \in R)))))$

```
DEFINITION FS.2.8: Infix function \circ. If BR[R] \wedge BR[S] then
R \circ S \simeq {<x,y> : (\exists z)(x \infixrl{R} z \wedge
z \infixrl{S} y)}. Otherwise R \circ S \up. Precedence 10.
```

DEFINITION FS.2.8: Infix function $\circ$. If `BR`$[R] \wedge$ `BR`$[S]$ then $R \circ S \simeq \{\langle x, y\rangle : (\exists z)(xRz \wedge zSy)\}$. Otherwise $R \circ S\uparrow$. Precedence 10.

$\circ(R, S) \simeq (\iota z_0)((\texttt{BR}[R] \wedge \texttt{BR}[S] \wedge z_0 \simeq (\iota y_0)(\forall x_0)(x_0 \in y_0 \leftrightarrow (\exists x, y)(x_0 = \varpi_0(x, y) \wedge ((\exists z)\, \varpi_0(x, z) \in R \wedge \varpi_0(z, y) \in S)))))$

```
DEFINITION FS.2.9: Infix function |. If BR[R] then R|A \simeq R \cap
(A \times Rng(R)). Otherwise R|A \up. Precedence 5.
```

DEFINITION FS.2.9: Infix function |. If $\texttt{BR}[R]$ then $R \mid A \simeq R \cap (A \times \texttt{Rng}(R))$. Otherwise $R \mid A\uparrow$. Precedence 5.

$|(R, A) \simeq (\iota x_0)((\texttt{BR}[R] \wedge x_0 \simeq \cap(R, (\times(A, \texttt{Rng}(R))))))$

```
DEFINITION FS.2.10: Infix function |_{Rng}. If BR[R] then R |_{Rng}
A \simeq Rng(R|A). Otherwise R |_{Rng} A \up. Precedence 5.
```

DEFINITION FS.2.10: Infix function $|_{\texttt{Rng}}$. If $\texttt{BR}[R]$ then $R \mid_{\texttt{Rng}} A \simeq \texttt{Rng}(R|A)$. Otherwise $R \mid_{\texttt{Rng}} A\uparrow$. Precedence 5.

$|_{\texttt{Rng}}(R, A) \simeq (\iota x_0)((\texttt{BR}[R] \wedge x_0 \simeq \texttt{Rng}(|(R, A))))$

```
DEFINITION FS.2.11: 2-ary relation RFX. RFX[R,A] \iff BR[R] \wedge
(\forall x \in A)(x \infixrl{R} x).
```

DEFINITION FS.2.11: 2-ary relation RFX. $\texttt{RFX}[R, A] \leftrightarrow \texttt{BR}[R] \wedge (\forall x \in A)(xRx)$.

$\texttt{RFX}[R, A] \leftrightarrow \texttt{BR}[R] \wedge (\forall x)(x \in A \rightarrow \varpi_0(x, x) \in R)$

```
DEFINITION FS.2.12: 2-ary relation IRFX. IRFX[R,A] \iff BR[R] \wedge
(\forall x \in A)(\neg(x \infixrl{R} x)).
```

DEFINITION FS.2.12: 2-ary relation IRFX. $\texttt{IRFX}[R, A] \leftrightarrow \texttt{BR}[R] \wedge (\forall x \in A)(\neg(xRx))$.

$\texttt{IRFX}[R, A] \leftrightarrow \texttt{BR}[R] \wedge (\forall x)(x \in A \rightarrow \neg(\varpi_0(x, x) \in R))$

```
DEFINITION FS.2.13: 2-ary relation SYM. SYM[R,A] \iff BR[R] \wedge
(\forall x,y \in A)(x \infixrl{R} y \iff y \infixrl{R} x).
```

DEFINITION FS.2.13: 2-ary relation SYM. $\texttt{SYM}[R, A] \leftrightarrow \texttt{BR}[R] \wedge (\forall x, y \in A)(xRy \leftrightarrow yRx)$.

$\texttt{SYM}[R, A] \leftrightarrow \texttt{BR}[R] \wedge (\forall x, y)(x \in A \wedge y \in A \rightarrow \varpi_0(x, y) \in R \leftrightarrow \varpi_0(y, x) \in R)$

```
DEFINITION FS.2.14: 2-ary relation ASYM. ASYM[R,A] \iff BR[R] \wedge
(\forall x,y \in A)(x \infixrl{R} y \implies \neg(y \infixrl{R} x)).
```

DEFINITION FS.2.14: 2-ary relation ASYM. $\texttt{ASYM}[R, A] \leftrightarrow \texttt{BR}[R] \wedge (\forall x, y \in A)(xRy \rightarrow \neg(yRx))$.

$\texttt{ASYM}[R, A] \leftrightarrow \texttt{BR}[R] \wedge (\forall x, y)(x \in A \wedge y \in A \rightarrow \varpi_0(x, y) \in R \rightarrow \neg(\varpi_0(y, x) \in R))$

```
DEFINITION FS.2.15: 2-ary relation ANSYM. ANSYM[R,A] \iff BR[R] \wedge
(\forall x,y \in A)(
x \infixrl{R} y \wedge y \infixrl{R} x \implies x = y
).
```

DEFINITION FS.2.15: 2-ary relation ANSYM. $\texttt{ANSYM}[R, A] \leftrightarrow \texttt{BR}[R] \wedge (\forall x, y \in A)(xRy \wedge yRx \rightarrow x = y)$.

$\texttt{ANSYM}[R, A] \leftrightarrow \texttt{BR}[R] \wedge (\forall x, y)(x \in A \wedge y \in A \rightarrow \varpi_0(x, y) \in R \wedge \varpi_0(y, x) \in R \rightarrow x = y)$

```
DEFINITION FS.2.16: 2-ary relation TRAN. TRAN[R,A] \iff BR[R] \wedge
(\forall x,y,z \in A)(x \infixrl{R} y \wedge y \infixrl{R} z
 \implies x \infixrl{R} z).
```

DEFINITION FS.2.16: 2-ary relation TRAN. $\texttt{TRAN}[R, A] \leftrightarrow \texttt{BR}[R] \wedge (\forall x, y, z \in A)(xRy \wedge yRz \rightarrow xRz)$.

$\texttt{TRAN}[R, A] \leftrightarrow \texttt{BR}[R] \wedge (\forall x, y, z)(x \in A \wedge y \in A \wedge z \in A \rightarrow \varpi_0(x, y) \in R \wedge \varpi_0(y, z) \in R \rightarrow \varpi_0(x, z) \in R)$

```
DEFINITION FS.2.17: 2-ary relation CONN. CONN[R,A] \iff BR[R] \wedge
(\forall x,y \in A)(
x \neq y \implies x \infixrl{R} y \vee y \infixrl{R} x
).
```

DEFINITION FS.2.17: 2-ary relation CONN. $\texttt{CONN}[R, A] \leftrightarrow \texttt{BR}[R] \wedge (\forall x, y \in A)(x \neq y \rightarrow xRy \vee yRx)$.

$\texttt{CONN}[R, A] \leftrightarrow \texttt{BR}[R] \wedge (\forall x, y)(x \in A \wedge y \in A \rightarrow \neg(x = y) \rightarrow \varpi_0(x, y) \in R \vee \varpi_0(y, x) \in R)$

```
DEFINITION FS.2.18: 2-ary relation SCONN. SCONN[R,A] \iff BR[R]
\wedge (\forall x,y \in A)(x \infixrl{R} y \vee y \infixrl{R} x).
```

DEFINITION FS.2.18: 2-ary relation SCONN. $\texttt{SCONN}[R, A] \leftrightarrow \texttt{BR}[R] \wedge (\forall x, y \in A)(xRy \vee yRx)$.

$\texttt{SCONN}[R, A] \leftrightarrow \texttt{BR}[R] \wedge (\forall x, y)(x \in A \wedge y \in A \rightarrow \varpi_0(x, y) \in R \vee \varpi_0(y, x) \in R)$

```
DEFINITION FS.2.19: 1-ary relation RFX. RFX[R] \iff BR[R] \wedge
RFX[R,Fld(R)].
```

DEFINITION FS.2.19: 1-ary relation RFX. $\texttt{RFX}[R] \leftrightarrow \texttt{BR}[R] \wedge \texttt{RFX}[R, \texttt{Fld}(R)]$.

$\texttt{RFX}[R] \leftrightarrow \texttt{BR}[R] \wedge \texttt{RFX}[R, \texttt{Fld}(R)]$

```
DEFINITION FS.2.20: 1-ary relation IRFX. IRFX[R] \iff BR[R] \wedge
IRFX[R,Fld(R)].
```

DEFINITION FS.2.20: 1-ary relation IRFX. $\texttt{IRFX}[R] \leftrightarrow \texttt{BR}[R] \wedge \texttt{IRFX}[R, \texttt{Fld}(R)]$.

$\texttt{IRFX}[R] \leftrightarrow \texttt{BR}[R] \wedge \texttt{IRFX}[R, \texttt{Fld}(R)]$

```
DEFINITION FS.2.21: 1-ary relation SYM. SYM[R] \iff BR[R] \wedge
SYM[R,Dom(R)].
```

DEFINITION FS.2.21: 1-ary relation SYM. $\texttt{SYM}[R] \leftrightarrow \texttt{BR}[R] \wedge \texttt{SYM}[R, \texttt{Dom}(R)]$.

$\texttt{SYM}[R] \leftrightarrow \texttt{BR}[R] \wedge \texttt{SYM}[R, \texttt{Dom}(R)]$

```
DEFINITION FS.2.22: 1-ary relation ASYM. ASYM[R] \iff BR[R] \wedge
ASYM[R,Dom(R)].
```

DEFINITION FS.2.22: 1-ary relation ASYM. $\texttt{ASYM}[R] \leftrightarrow \texttt{BR}[R] \wedge \texttt{ASYM}[R, \texttt{Dom}(R)]$.

$\texttt{ASYM}[R] \leftrightarrow \texttt{BR}[R] \wedge \texttt{ASYM}[R, \texttt{Dom}(R)]$

```
DEFINITION FS.2.23: 1-ary relation ANSYM. ANSYM[R] \iff BR[R] \wedge
ANSYM[R,Dom(R)].
```

DEFINITION FS.2.23: 1-ary relation ANSYM. $\texttt{ANSYM}[R] \leftrightarrow \texttt{BR}[R] \wedge \texttt{ANSYM}[R, \texttt{Dom}(R)]$.

$\texttt{ANSYM}[R] \leftrightarrow \texttt{BR}[R] \wedge \texttt{ANSYM}[R, \texttt{Dom}(R)]$

```
DEFINITION FS.2.24: 1-ary relation TRAN. TRAN[R] \iff BR[R] \wedge
TRAN[R,Dom(R)].
```

DEFINITION FS.2.24: 1-ary relation TRAN. $\texttt{TRAN}[R] \leftrightarrow \texttt{BR}[R] \wedge \texttt{TRAN}[R, \texttt{Dom}(R)]$.

$\texttt{TRAN}[R] \leftrightarrow \texttt{BR}[R] \wedge \texttt{TRAN}[R, \texttt{Dom}(R)]$

```
DEFINITION FS.2.25: 1-ary relation CONN. CONN[R] \iff BR[R] \wedge
CONN[R,Dom(R)].
```

DEFINITION FS.2.25: 1-ary relation CONN. $\texttt{CONN}[R] \leftrightarrow \texttt{BR}[R] \wedge \texttt{CONN}[R, \texttt{Dom}(R)]$.

$\texttt{CONN}[R] \leftrightarrow \texttt{BR}[R] \wedge \texttt{CONN}[R, \texttt{Dom}(R)]$

```
DEFINITION FS.2.26: 1-ary relation SCONN. SCONN[R] \iff BR[R] \wedge
SCONN[R,Dom(R)].
```

DEFINITION FS.2.26: 1-ary relation SCONN. $\texttt{SCONN}[R] \leftrightarrow \texttt{BR}[R] \wedge \texttt{SCONN}[R, \texttt{Dom}(R)]$.

$\texttt{SCONN}[R] \leftrightarrow \texttt{BR}[R] \wedge \texttt{SCONN}[R, \texttt{Dom}(R)]$

```
DEFINITION FS.2.27: 1-ary function Id. Id(x) \simeq {<y,y> : y \in x}.
```

DEFINITION FS.2.27: 1-ary function Id. $\texttt{Id}(x) \simeq \{\langle y, y\rangle : y \in x\}$.

$\texttt{Id}(x) \simeq (\iota y_0)(\forall x_0)(x_0 \in y_0 \leftrightarrow (\exists y)(x_0 = \varpi_0(y, y) \wedge (y \in x)))$

```
DEFINITION FS.2.28: 2-ary relation QORD. QORD[R,A] \iff RFX[R,A] \wedge
TRAN[R,A].
```

DEFINITION FS.2.28: 2-ary relation QORD. $\texttt{QORD}[R, A] \leftrightarrow \texttt{RFX}[R, A] \wedge \texttt{TRAN}[R, A]$.

$\texttt{QORD}[R, A] \leftrightarrow \texttt{RFX}[R, A] \wedge \texttt{TRAN}[R, A]$

```
DEFINITION FS.2.29: 2-ary relation PORD. PORD[R,A] \iff RFX[R,A]
\wedge ANSYM[R,A] \wedge TRAN[R,A].
```

DEFINITION FS.2.29: 2-ary relation PORD. $\texttt{PORD}[R, A] \leftrightarrow \texttt{RFX}[R, A] \wedge \texttt{ANSYM}[R, A] \wedge \texttt{TRAN}[R, A]$.

$\texttt{PORD}[R, A] \leftrightarrow \texttt{RFX}[R, A] \wedge \texttt{ANSYM}[R, A] \wedge \texttt{TRAN}[R, A]$

```
DEFINITION FS.2.30: 2-ary relation SORD. SORD[R,A] \iff ANSYM[R,A]
\wedge TRAN[R,A] \wedge SCONN[R,A].
```

DEFINITION FS.2.30: 2-ary relation SORD. $\texttt{SORD}[R, A] \leftrightarrow \texttt{ANSYM}[R, A] \wedge \texttt{TRAN}[R, A] \wedge \texttt{SCONN}[R, A]$.

$\texttt{SORD}[R, A] \leftrightarrow \texttt{ANSYM}[R, A] \wedge \texttt{TRAN}[R, A] \wedge \texttt{SCONN}[R, A]$

```
DEFINITION FS.2.31: 2-ary relation SPORD. SPORD[R,A] \iff ASYM[R,A]
\wedge TRAN[R,A].
```

DEFINITION FS.2.31: 2-ary relation SPORD. $\texttt{SPORD}[R, A] \leftrightarrow \texttt{ASYM}[R, A] \wedge \texttt{TRAN}[R, A]$.

$\texttt{SPORD}[R, A] \leftrightarrow \texttt{ASYM}[R, A] \wedge \texttt{TRAN}[R, A]$

```
DEFINITION FS.2.32: 2-ary relation SSORD. SSORD[R,A] \iff ASYM[R,A]
\wedge TRAN[R,A] \wedge CONN[R,A].
```

DEFINITION FS.2.32: 2-ary relation SSORD. $\texttt{SSORD}[R, A] \leftrightarrow \texttt{ASYM}[R, A] \wedge \texttt{TRAN}[R, A] \wedge \texttt{CONN}[R, A]$.

$\texttt{SSORD}[R, A] \leftrightarrow \texttt{ASYM}[R, A] \wedge \texttt{TRAN}[R, A] \wedge \texttt{CONN}[R, A]$

```
DEFINITION FS.2.33: 1-ary relation QORD. QORD[R] \iff QORD[R,Fld(R)].
```

DEFINITION FS.2.33: 1-ary relation QORD. $\texttt{QORD}[R] \leftrightarrow \texttt{QORD}[R, \texttt{Fld}(R)]$.

$\texttt{QORD}[R] \leftrightarrow \texttt{QORD}[R, \texttt{Fld}(R)]$

```
DEFINITION FS.2.34: 1-ary relation PORD. PORD[R] \iff PORD[R,Fld(R)].
```

DEFINITION FS.2.34: 1-ary relation PORD. PORD$[R] \leftrightarrow$ PORD$[R,$ Fld$(R)]$.

PORD$[R] \leftrightarrow$ PORD$[R,$ Fld$(R)]$

```
DEFINITION FS.2.35: 1-ary relation SORD. SORD[R] \iff SORD[R,Fld(R)].
```

DEFINITION FS.2.35: 1-ary relation SORD. SORD$[R] \leftrightarrow$ SORD$[R,$ Fld$(R)]$.

SORD$[R] \leftrightarrow$ SORD$[R,$ Fld$(R)]$

```
DEFINITION FS.2.36: 1-ary relation SPORD. SPORD[R] \iff
SPORD[R,Fld(R)].
```

DEFINITION FS.2.36: 1-ary relation SPORD. SPORD$[R] \leftrightarrow$ SPORD$[R,$ Fld$(R)]$.

SPORD$[R] \leftrightarrow$ SPORD$[R,$ Fld$(R)]$

```
DEFINITION FS.2.37: 1-ary relation SSORD. SSORD[R] \iff
SSORD[R,Fld(R)].
```

DEFINITION FS.2.37: 1-ary relation SSORD. SSORD$[R] \leftrightarrow$ SSORD$[R,$ Fld$(R)]$.

SSORD$[R] \leftrightarrow$ SSORD$[R,$ Fld$(R)]$

```
DEFINITION FS.2.38: 3-ary relation MELT. MELT[x,R,A] \iff BR[R]
\wedge x \in A \wedge (\forall y \in A)(\neg(y \infixrl{R} x)).
```

DEFINITION FS.2.38: 3-ary relation MELT. MELT$[x, R, A] \leftrightarrow$ BR$[R] \wedge x \in A \wedge (\forall y \in A)$ $(\neg(yRx))$.

MELT$[x, R, A] \leftrightarrow$ BR$[R] \wedge x \in A \wedge (\forall y)(y \in A \rightarrow \neg(\varpi_0(y, x) \in R))$

```
DEFINITION FS.2.39: 3-ary relation FELT. FELT[x,R,A] \iff BR[R] \wedge
x \in A \wedge (\forall y \in A)(x \neq y \implies x \infixrl{R} y).
```

DEFINITION FS.2.39: 3-ary relation FELT. $\texttt{FELT}[x, R, A] \leftrightarrow \texttt{BR}[R] \wedge x \in A \wedge (\forall y \in A)(x \neq y \rightarrow xRy)$.

$\texttt{FELT}[x, R, A] \leftrightarrow \texttt{BR}[R] \wedge x \in A \wedge (\forall y)(y \in A \rightarrow \neg(x = y) \rightarrow \varpi_0(x, y) \in R)$

```
DEFINITION FS.2.40: 2-ary relation WO. WO[R,A] \iff CONN[R,A] \wedge
(\forall B \subseteq A)(
  B \neq \varnothing \implies (\exists x)(MELT[x,R,B])
).
```

DEFINITION FS.2.40: 2-ary relation WO. $\texttt{WO}[R, A] \leftrightarrow \texttt{CONN}[R, A] \wedge (\forall B \subseteq A)(B \neq \varnothing \rightarrow (\exists x)(\texttt{MELT}[x, R, B]))$.

$\texttt{WO}[R, A] \leftrightarrow \texttt{CONN}[R, A] \wedge (\forall B)(\subseteq [B, A] \rightarrow \neg(B = \varnothing) \rightarrow (\exists x)\, \texttt{MELT}[x, R, B])$

```
DEFINITION FS.2.41: 3-ary relation ISUC. ISUC[y,R,x] \iff BR[R]
\wedge x \infixrl{R} y \wedge (\forall z)(
  x \infixrl{R} z \implies z = y \vee y \infixrl{R} z
).
```

DEFINITION FS.2.41: 3-ary relation ISUC. $\texttt{ISUC}[y, R, x] \leftrightarrow \texttt{BR}[R] \wedge xRy \wedge (\forall z)(xRz \rightarrow z = y \vee yRz)$.

$\texttt{ISUC}[y, R, x] \leftrightarrow \texttt{BR}[R] \wedge \varpi_0(x, y) \in R \wedge (\forall z)\, \varpi_0(x, z) \in R \rightarrow z = y \vee \varpi_0(y, z) \in R$

```
DEFINITION FS.2.42: 3-ary relation LELT. LELT[x,R,A] \iff BR[R] \wedge
x \in A \wedge (\forall y \in A)(x \neq y \implies y \infixrl{R} x).
```

DEFINITION FS.2.42: 3-ary relation LELT. $\texttt{LELT}[x, R, A] \leftrightarrow \texttt{BR}[R] \wedge x \in A \wedge (\forall y \in A)(x \neq y \rightarrow yRx)$.

$\texttt{LELT}[x, R, A] \leftrightarrow \texttt{BR}[R] \wedge x \in A \wedge (\forall y)(y \in A \rightarrow \neg(x = y) \rightarrow \varpi_0(y, x) \in R)$

```
DEFINITION FS.2.43: 3-ary relation SECT. SECT[B,R,A] \iff BR[R] \wedge
B \subseteq A \wedge A \cap Cnv(R) |_{Rng} B \subseteq B.
```

DEFINITION FS.2.43: 3-ary relation SECT. $\texttt{SECT}[B, R, A] \leftrightarrow \texttt{BR}[R] \wedge B \subseteq A \wedge A \cap \texttt{Cnv}(R)|_{Rng} B \subseteq B$.

$\texttt{SECT}[B, R, A] \leftrightarrow \texttt{BR}[R] \wedge \subseteq [B, A] \wedge \subseteq [\cap (A, |_{\texttt{Rng}}(\texttt{Cnv}(R), B)), B]$

```
DEFINITION FS.2.44: 3-ary function Seg. If BR[R] then Seg(A,R,x)
\simeq {y \in A : y \infixrl{R} x}. Otherwise Seg(R) \up.
```

DEFINITION FS.2.44: 3-ary function Seg. If $\texttt{BR}[R]$ then $\texttt{Seg}(A, R, x) \simeq \{y \in A : yRx\}$. Otherwise $\texttt{Seg}(R)\uparrow$.

$\texttt{Seg}(A, R, x) \simeq (\iota z_0)((\texttt{BR}[R] \wedge z_0 \simeq (\iota y_0)(\forall x_0)(x_0 \in y_0 \leftrightarrow (\exists y)(x_0 = y \wedge (\varpi_0(y, x) \in R) \wedge (y \in A)))))$

```
DEFINITION FS.2.45: 3-ary relation LB. LB[x,R,A] \iff BR[R] \wedge
(\forall y \in A)(x \infixrl{R} y).
```

DEFINITION FS.2.45: 3-ary relation LB. $\texttt{LB}[x, R, A] \leftrightarrow \texttt{BR}[R] \wedge (\forall y \in A)(xRy)$.

$\texttt{LB}[x, R, A] \leftrightarrow \texttt{BR}[R] \wedge (\forall y)(y \in A \rightarrow \varpi_0(x, y) \in R)$

```
DEFINITION FS.2.46: 3-ary relation INF. INF[x,R,A] \iff LB[x,R,A]
\wedge (\forall y \in A)(LB[y,R,A] \implies y \infixrl{R} x).
```

DEFINITION FS.2.46: 3-ary relation INF. $\texttt{INF}[x, R, A] \leftrightarrow \texttt{LB}[x, R, A] \wedge (\forall y \in A)(\texttt{LB}[y, R, A] \rightarrow yRx)$.

$\texttt{INF}[x, R, A] \leftrightarrow \texttt{LB}[x, R, A] \wedge (\forall y)(y \in A \rightarrow \texttt{LB}[y, R, A] \rightarrow \varpi_0(y, x) \in R)$

```
DEFINITION FS.2.47: 3-ary relation UB. UB[x,R,A] \iff BR[R] \wedge
(\forall y \in A)(y \infixrl{R} x).
```

DEFINITION FS.2.47: 3-ary relation UB. $\texttt{UB}[x, R, A] \leftrightarrow \texttt{BR}[R] \wedge (\forall y \in A)(yRx)$.

$\texttt{UB}[x, R, A] \leftrightarrow \texttt{BR}[R] \wedge (\forall y)(y \in A \rightarrow \varpi_0(y, x) \in R)$

```
DEFINITION FS.2.48: 3-ary relation SUP. SUP[x,R,A] \iff UB[x,R,A]
\wedge (\forall y \in A)(UB[y,R,A] \implies x \infixrl{R} y).
```

DEFINITION FS.2.48: 3-ary relation SUP. $\texttt{SUP}[x, R, A] \leftrightarrow \texttt{UB}[x, R, A] \wedge (\forall y \in A)(\texttt{UB}[y, R, A] \rightarrow xRy)$.

$\texttt{SUP}[x, R, A] \leftrightarrow \texttt{UB}[x, R, A] \wedge (\forall y)(y \in A \rightarrow \texttt{UB}[y, R, A] \rightarrow \varpi_0(x, y) \in R)$

```
DEFINITION FS.2.50: 1-ary relation EQUIV. EQUIV[R] \iff RFX[R] \wedge
SYM[R] \wedge TRAN[R].
```

DEFINITION FS.2.50: 1-ary relation EQUIV. $\texttt{EQUIV}[R] \leftrightarrow \texttt{RFX}[R] \wedge \texttt{SYM}[R] \wedge \texttt{TRAN}[R]$.

$\texttt{EQUIV}[R] \leftrightarrow \texttt{RFX}[R] \wedge \texttt{SYM}[R] \wedge \texttt{TRAN}[R]$

```
DEFINITION FS.2.51: 2-ary relation EQUIV. EQUIV[R,A] \iff EQUIV[R]
\wedge Fld(R) = A.
```

DEFINITION FS.2.51: 2-ary relation EQUIV. $\texttt{EQUIV}[R, A] \leftrightarrow \texttt{EQUIV}[R] \wedge \texttt{Fld}(R) = A$.

$\texttt{EQUIV}[R, A] \leftrightarrow \texttt{EQUIV}[R] \wedge \texttt{Fld}(R) = A$

```
DEFINITION FS.2.52: 2-ary function Coset. If EQUIV[R] \wedge
x \in Fld(R) then Coset(x,R) \simeq {y : x \infixrl{R} y}.
Otherwise Coset(x,R) \up.
```

DEFINITION FS.2.52: 2-ary function Coset. If EQUIV$[R] \wedge x \in$ Fld(R) then Coset$(x, R) \simeq \{y : xRy\}$. Otherwise Coset$(x, R)\uparrow$.

Coset$(x, R) \simeq (\iota z_0)(($EQUIV$[R] \wedge x \in$ Fld$(R) \wedge z_0 \simeq (\iota y_0)(\forall x_0)(x_0 \in y_0 \leftrightarrow (\exists y)(x_0 = y \wedge (\varpi_0(x, y) \in R)))))$

```
DEFINITION FS.2.53: 2-ary relation PART. PART[W,A] \iff
\cup(W) = A \wedge (\forall B,C \in W)(
B \neq C \implies B \cap C = \varnothing
) \wedge
(\forall B \in W)(B \neq \varnothing).
```

DEFINITION FS.2.53: 2-ary relation PART. PART$[W, A] \leftrightarrow \cup(W) = A \wedge (\forall B, C \in W)(B \neq C \rightarrow B \cap C = \varnothing) \wedge (\forall B \in W)(B \neq \varnothing)$.

PART$[W, A] \leftrightarrow \cup(W) = A \wedge (\forall B, C)(B \in W \wedge C \in W \rightarrow \neg(B = C) \rightarrow \cap(B, C) = \varnothing) \wedge (\forall B)(B \in W \rightarrow \neg(B = \varnothing))$

```
DEFINITION FS.2.54: 1-ary relation PART. PART[W] \iff
(\exists A)(PART[W,A]).
```

DEFINITION FS.2.54: 1-ary relation PART. PART$[W] \leftrightarrow (\exists A)($PART$[W, A])$.

PART$[W] \leftrightarrow (\exists A)$ PART$[W, A]$

```
DEFINITION FS.2.55: 2-ary relation FINER. If PART[V] \wedge PART[W]
then FINER[V,W] \iff V \neq W \wedge
(\forall A \in V)(\exists B \in W)(A \subseteq B).
```

DEFINITION FS.2.55: 2-ary relation FINER. If PART$[V] \wedge$PART$[W]$ then FINER$[V, W] \leftrightarrow V \neq W \wedge (\forall A \in V)(\exists B \in W)(A \subseteq B)$.

FINER$[V, W] \leftrightarrow ($PART$[V] \wedge$ PART$[W] \wedge \neg(V = W) \wedge (\forall A)(A \in V \rightarrow (\exists B)(B \in W \wedge \subseteq[A, B])))$

```
DEFINITION FS.2.56: 1-ary function Part. If EQUIV[R] then Part(R)
\simeq {Coset(x,R) : x \in Fld(R)}.
```

DEFINITION FS.2.56: 1-ary function `Part`. If `EQUIV`$[R]$ then `Part`$(R) \simeq \{$`Coset`$(x, R) : x \in$ `Fld`$(R)\}$.

`Part`$(R) \simeq (\iota z_0)($`EQUIV`$[R] \wedge z_0 \simeq (\iota y_0)(\forall x_0)(x_0 \in y_0 \leftrightarrow (\exists x, R)(x_0 =$ `Coset`$(x, R) \wedge (x \in$ `Fld`$(R)))))$

```
DEFINITION FS.2.57: 1-ary function Reln. If PART[W] then Reln(W)
\simeq {
  <x,y> : (\exists B \in W)(x \in B \wedge y \in B)
}.
```

DEFINITION FS.2.57: 1-ary function `Reln`. If `PART`$[W]$ then `Reln`$(W) \simeq \{\langle x, y\rangle : (\exists B \in W)(x \in B \wedge y \in B)\}$.

`Reln`$(W) \simeq (\iota z_0)($`PART`$[W] \wedge z_0 \simeq (\iota y_0)(\forall x_0)(x_0 \in y_0 \leftrightarrow (\exists x, y)(x_0 = \varpi_0(x, y) \wedge ((\exists B)(B \in W \wedge x \in B \wedge y \in B)))))$

```
DEFINITION FS.2.58: 1-ary relation FCN. FCN[f] \iff
f = {<x,y> : f(x) = y}.
```

DEFINITION FS.2.58: 1-ary relation `FCN`. `FCN`$[f] \leftrightarrow f = \{\langle x, y\rangle : f(x) = y\}$.

`FCN`$[f] \leftrightarrow f = (\iota z_0)(\forall y_0)(y_0 \in z_0 \leftrightarrow (\exists x, y)(y_0 = \varpi_0(x, y) \wedge ((\iota x_0)(\varpi_0(x, x_0) \in f) = y)))$

```
DEFINITION FS.2.59: 1-ary relation MONO. MONO[f] \iff FCN[f] \wedge
FCN[Cnv(f)].
```

DEFINITION FS.2.59: 1-ary relation `MONO`. `MONO`$[f] \leftrightarrow$ `FCN`$[f] \wedge$ `FCN`$[$`Cnv`$(f)]$.

`MONO`$[f] \leftrightarrow$ `FCN`$[f] \wedge$ `FCN`$[$`Cnv`$(f)]$

```
DEFINITION FS.2.60: 3-ary relation FCN. FCN[f,A,B] \iff FCN[f] \wedge
Dom(f) = A \wedge Rng(f) \subseteq B.
```

DEFINITION FS.2.60: 3-ary relation FCN. $\texttt{FCN}[f, A, B] \leftrightarrow \texttt{FCN}[f] \wedge \texttt{Dom}(f) = A \wedge \texttt{Rng}(f) \subseteq B$.

$\texttt{FCN}[f, A, B] \leftrightarrow \texttt{FCN}[f] \wedge \texttt{Dom}(f) = A \wedge \subseteq[\texttt{Rng}(f), B]$

```
DEFINITION FS.2.61: 3-ary relation SURJ. SURJ[f,A,B] \iff FCN[f]
\wedge Dom(f) = A \wedge Rng(f) = B.
```

DEFINITION FS.2.61: 3-ary relation SURJ. $\texttt{SURJ}[f, A, B] \leftrightarrow \texttt{FCN}[f] \wedge \texttt{Dom}(f) = A \wedge \texttt{Rng}(f) = B$.

$\texttt{SURJ}[f, A, B] \leftrightarrow \texttt{FCN}[f] \wedge \texttt{Dom}(f) = A \wedge \texttt{Rng}(f) = B$

```
DEFINITION FS.2.62: 3-ary relation MONO. MONO[f,A,B] \iff MONO[f]
\wedge Dom(f) = A \wedge Rng(f) \subseteq B.
```

DEFINITION FS.2.62: 3-ary relation MONO. $\texttt{MONO}[f, A, B] \leftrightarrow \texttt{MONO}[f] \wedge \texttt{Dom}(f) = A \wedge \texttt{Rng}(f) \subseteq B$.

$\texttt{MONO}[f, A, B] \leftrightarrow \texttt{MONO}[f] \wedge \texttt{Dom}(f) = A \wedge \subseteq[\texttt{Rng}(f), B]$

```
DEFINITION FS.2.63: 3-ary relation BIJ. BIJ[f,A,B] \iff MONO[f]
\wedge Dom(f) = A \wedge Rng(f) = B.
```

DEFINITION FS.2.63: 3-ary relation BIJ. $\texttt{BIJ}[f, A, B] \leftrightarrow \texttt{MONO}[f] \wedge \texttt{Dom}(f) = A \wedge \texttt{Rng}(f) = B$.

$\texttt{BIJ}[f, A, B] \leftrightarrow \texttt{MONO}[f] \wedge \texttt{Dom}(f) = A \wedge \texttt{Rng}(f) = B$

```
DEFINITION FS.2.64: 2-ary function Maps. Maps(A,B) \simeq
{f : FCN[f,A,B]}.
```

DEFINITION FS.2.64: 2-ary function Maps. $\texttt{Maps}(A, B) \simeq \{f : \texttt{FCN}[f, A, B]\}$.

$\texttt{Maps}(A, B) \simeq (\iota y_0)(\forall x_0)(x_0 \in y_0 \leftrightarrow (\exists f)(x_0 = f \wedge (\texttt{FCN}[f, A, B])))$

F.3 Chapter 4

```
DEFINITION FS.3.1: Infix relation \approx_{C}. A \approx_{C} B \iff
(\exists f)(BIJ[f,A,B]).
```

DEFINITION FS.3.1: Infix relation $\approx_{\mathtt{C}}$. $A \approx_{\mathtt{C}} B \leftrightarrow (\exists f)(\mathtt{BIJ}[f, A, B])$.

$\approx_{\mathtt{C}}[A, B] \leftrightarrow (\exists f)\, \mathtt{BIJ}[f, A, B]$

```
DEFINITION FS.3.2: Infix relation \leq_{C}. x \leq_{C} y \iff
(\exists z \subseteq y)(x \approx_{C} z).
```

DEFINITION FS.3.2: Infix relation $\leq_{\mathtt{C}}$. $x \leq_{\mathtt{C}} y \leftrightarrow (\exists z \subseteq y)(x \approx_{\mathtt{C}} z)$.

$\leq_{\mathtt{C}}[x, y] \leftrightarrow (\exists z)(\subseteq[z, y] \wedge \approx_{\mathtt{C}}[x, z])$

```
DEFINITION FS.3.3: Infix relation <_{C}. A <_{C} B \iff A \leq_{C} B
\wedge \neg (B \leq_{C} A).
```

DEFINITION FS.3.3: Infix relation $<_{\mathtt{C}}$. $A <_{\mathtt{C}} B \leftrightarrow A \leq_{\mathtt{C}} B \wedge \neg(B \leq_{\mathtt{C}} A)$.

$<_{\mathtt{C}}[A, B] \leftrightarrow \leq_{\mathtt{C}}[A, B] \wedge \neg(\leq_{\mathtt{C}}[B, A])$

```
DEFINITION FS.3.4: 2-ary relation MNEL. MNEL[x,A] \iff x \in A \wedge
(\forall y \in A)(\neg(y \in x)).
```

DEFINITION FS.3.4: 2-ary relation MNEL. $\mathtt{MNEL}[x, A] \leftrightarrow x \in A \wedge (\forall y \in A)(\neg(y \in x))$.

$\mathtt{MNEL}[x, A] \leftrightarrow x \in A \wedge (\forall y)(y \in A \rightarrow \neg(y \in x))$

```
DEFINITION FS.3.5: 2-ary relation MXEL. MXEL[x,A] \iff x \in A \wedge
(\forall y \in A)(\neg(x \in y)).
```

DEFINITION FS.3.5: 2-ary relation MXEL. $\texttt{MXEL}[x, A] \leftrightarrow x \in A \wedge (\forall y \in A)(\neg(x \in y))$.

$\texttt{MXEL}[x, A] \leftrightarrow x \in A \wedge (\forall y)(y \in A \rightarrow \neg(x \in y))$

```
DEFINITION FS.3.6: 1-ary relation FIN. FIN[x] \iff
(\forall A \neq \varnothing)(
  A \subseteq \wp(x) \implies (\exists y \in A)(MNEL[y,A])
).
```

DEFINITION FS.3.6: 1-ary relation FIN. $\texttt{FIN}[x] \leftrightarrow (\forall A \neq \varnothing)(A \subseteq \wp(x) \rightarrow (\exists y \in A)(\texttt{MNEL}[y, A]))$.

$\texttt{FIN}[x] \leftrightarrow (\forall A)(\neg(A = \varnothing) \rightarrow \subseteq[A, \wp(x)] \rightarrow (\exists y)(y \in A \wedge \texttt{MNEL}[y, A]))$

```
DEFINITION FS.3.7: 1-ary relation DFIN. DFIN[x] \iff
(\forall y \subseteq x)(
  y \neq x \implies \neg(x \approx_{C} y)
).
```

DEFINITION FS.3.7: 1-ary relation DFIN. $\texttt{DFIN}[x] \leftrightarrow (\forall y \subseteq x)(y \neq x \rightarrow \neg(x \approx_{\mathtt{C}} y))$.

$\texttt{DFIN}[x] \leftrightarrow (\forall y)(\subseteq[y, x] \rightarrow \neg(y = x) \rightarrow \neg(\approx_{\mathtt{C}}[x, y]))$

F.4 Chapter 5

```
DEFINITION FS.4.1: 1-ary relation TRANS. TRANS[x] \iff
(\forall y \in x)(\forall z \in y)(z \in x).
```

DEFINITION FS.4.1: 1-ary relation TRANS. $\texttt{TRANS}[x] \leftrightarrow (\forall y \in x)(\forall z \in y)(z \in x)$.

$\texttt{TRANS}[x] \leftrightarrow (\forall y)(y \in x \rightarrow (\forall z)(z \in y \rightarrow z \in x))$

```
DEFINITION FS.4.2: 1-ary relation ECONN. ECONN[x] \iff
(\forall y,z \in x)(
  y \in z \vee z \in y \vee y = z
).
```

DEFINITION FS.4.2: 1-ary relation ECONN. $\texttt{ECONN}[x] \leftrightarrow (\forall y, z \in x)(y \in z \vee z \in y \vee y = z)$.

$\texttt{ECONN}[x] \leftrightarrow (\forall y, z)(y \in x \wedge z \in x \rightarrow y \in z \vee z \in y \vee y = z)$

```
DEFINITION FS.4.3: 1-ary relation ORD. ORD[x] \iff
TRANS[x] \wedge ECONN[x].
```

DEFINITION FS.4.3: 1-ary relation ORD. $\texttt{ORD}[x] \leftrightarrow \texttt{TRANS}[x] \wedge \texttt{ECONN}[x]$.

$\texttt{ORD}[x] \leftrightarrow \texttt{TRANS}[x] \wedge \texttt{ECONN}[x]$

```
DEFINITION FS.4.4: 1-ary function Eps. Eps(x) \simeq {
  <y,z> : y \in z \wedge z,y \in x
}.
```

DEFINITION FS.4.4: 1-ary function Eps. $\texttt{Eps}(x) \simeq \{\langle y, z\rangle : y \in z \wedge z, y \in x\}$.

$\texttt{Eps}(x) \simeq (\iota y_0)(\forall x_0)(x_0 \in y_0 \leftrightarrow (\exists y, z)(x_0 = \varpi_0(y, z) \wedge (y \in z \wedge z \in x \wedge y \in x)))$

```
DEFINITION FS.4.5: Infix relation <_{O}. A <_{O} B \iff
ORD[A] \wedge ORD[B] \wedge A \in B.
```

DEFINITION FS.4.5: Infix relation $<_0$. $A <_0 B \leftrightarrow \texttt{ORD}[A] \wedge \texttt{ORD}[B] \wedge A \in B$.

$<_0[A, B] \leftrightarrow \texttt{ORD}[A] \wedge \texttt{ORD}[B] \wedge A \in B$

```
DEFINITION FS.4.6: Infix relation \leq_{O}. A \leq_{O} B \iff
ORD[A] \wedge ORD[B] \wedge (A \in B \vee A = B).
```

DEFINITION FS.4.6: Infix relation $\leq_{\mathtt{O}}$. $A \leq_{\mathtt{O}} B \leftrightarrow \mathtt{ORD}[A] \wedge \mathtt{ORD}[B] \wedge (A \in B \vee A = B)$.

$\leq_{\mathtt{O}}[A, B] \leftrightarrow \mathtt{ORD}[A] \wedge \mathtt{ORD}[B] \wedge A \in B \vee A = B$

```
DEFINITION FS.4.7: Infix relation >_{O}. A >_{O} B \iff
ORD[A] \wedge ORD[B] \wedge B \in A.
```

DEFINITION FS.4.7: Infix relation $>_{\mathtt{O}}$. $A >_{\mathtt{O}} B \leftrightarrow \mathtt{ORD}[A] \wedge \mathtt{ORD}[B] \wedge B \in A$.

$>_{\mathtt{O}}[A, B] \leftrightarrow \mathtt{ORD}[A] \wedge \mathtt{ORD}[B] \wedge B \in A$

```
DEFINITION FS.4.8: Infix relation \geq_{O}. A \geq_{O} B \iff
ORD[A] \wedge ORD[B] \wedge (B \in A \vee A = B).
```

DEFINITION FS.4.8: Infix relation $\geq_{\mathtt{O}}$. $A \geq_{\mathtt{O}} B \leftrightarrow \mathtt{ORD}[A] \wedge \mathtt{ORD}[B] \wedge (B \in A \vee A = B)$.

$\geq_{\mathtt{O}}[A, B] \leftrightarrow \mathtt{ORD}[A] \wedge \mathtt{ORD}[B] \wedge B \in A \vee A = B$

```
DEFINITION FS.4.9: 1-ary function Suc. If ORD[x] then Suc(x) \simeq
{y: y \leq_{O} x}. Otherwise Suc(x) \up.
```

DEFINITION FS.4.9: 1-ary function $\mathtt{Suc}$. If $\mathtt{ORD}[x]$ then $\mathtt{Suc}(x) \simeq \{y : y \leq_{\mathtt{O}} x\}$. Otherwise $\mathtt{Suc}(x)\uparrow$.

$\mathtt{Suc}(x) \simeq (\iota z_0)((\mathtt{ORD}[x] \wedge z_0 \simeq (\iota y_0)(\forall x_0)(x_0 \in y_0 \leftrightarrow (\exists y)(x_0 = y \wedge (\leq_{\mathtt{O}}[y, x])))))$

```
DEFINITION FS.4.10: 1-ary relation NAT. NAT[x] \iff ORD[x] \wedge
WO[Cnv(Eps(x)),x].
```

DEFINITION FS.4.10: 1-ary relation NAT. $\texttt{NAT}[x] \leftrightarrow \texttt{ORD}[x] \wedge \texttt{WO}[\texttt{Cnv}(\texttt{Eps}(x)), x]$.

$\texttt{NAT}[x] \leftrightarrow \texttt{ORD}[x] \wedge \texttt{WO}[\texttt{Cnv}(\texttt{Eps}(x)), x]$

```
DEFINITION FS.4.11: 0-ary function \omega. \omega \simeq {x : NAT[x]}.
```

DEFINITION FS.4.11: 0-ary function ω. $\omega \simeq \{x : \texttt{NAT}[x]\}$.

$\omega \simeq (\iota y_0)(\forall x_0)(x_0 \in y_0 \leftrightarrow (\exists x)(x_0 = x \wedge (\texttt{NAT}[x])))$

```
DEFINITION FS.4.11.a: 0-ary function \mathbb{N}. \mathbb{N}
\simeq \omega.
```

DEFINITION FS.4.11.a: 0-ary function $\mathbb{N}$. $\mathbb{N} \simeq \omega$.

$\mathbb{N} \simeq \omega$

```
DEFINITION FS.4.12: 0-ary function 0_{N}. 0_{N} \simeq \varnothing.
```

DEFINITION FS.4.12: 0-ary function $\texttt{0}_{\texttt{N}}$. $\texttt{0}_{\texttt{N}} \simeq \varnothing$.

$\texttt{0}_{\texttt{N}} \simeq \varnothing$

```
DEFINITION FS.4.13: 0-ary function 1_{N}. 1_{N} \simeq {\varnothing}.
```

DEFINITION FS.4.13: 0-ary function $\texttt{1}_{\texttt{N}}$. $\texttt{1}_{\texttt{N}} \simeq \{\varnothing\}$.

$\texttt{1}_{\texttt{N}} \simeq (\iota x_0)(\forall y_0)(y_0 \in x_0 \leftrightarrow y_0 = \varnothing)$

```
DEFINITION FS.4.13.2: 0-ary function 2_{N}. 2_{N} \simeq Suc(1_{N}).
```

DEFINITION FS.4.13.2: 0-ary function 2_N. $2_N \simeq \mathtt{Suc}(1_N)$.

$2_N \simeq \mathtt{Suc}(1_N)$

```
DEFINITION FS.4.13.3: 0-ary function 3_{N}. 3_{N} \simeq Suc(2_{N}).
```

DEFINITION FS.4.13.3: 0-ary function 3_N. $3_N \simeq \mathtt{Suc}(2_N)$.

$3_N \simeq \mathtt{Suc}(2_N)$

```
DEFINITION FS.4.13.4: 0-ary function 4_{N}. 4_{N} \simeq Suc(3_{N}).
```

DEFINITION FS.4.13.4: 0-ary function 4_N. $4_N \simeq \mathtt{Suc}(3_N)$.

$4_N \simeq \mathtt{Suc}(3_N)$

```
DEFINITION FS.4.13.5: 0-ary function 5_{N}. 5_{N} \simeq Suc(4_{N}).
```

DEFINITION FS.4.13.5: 0-ary function 5_N. $5_N \simeq \mathtt{Suc}(4_N)$.

$5_N \simeq \mathtt{Suc}(4_N)$

```
DEFINITION FS.4.13.6: 0-ary function 6_{N}. 6_{N} \simeq Suc(5_{N}).
```

DEFINITION FS.4.13.6: 0-ary function 6_N. $6_N \simeq \mathtt{Suc}(5_N)$.

$6_N \simeq \mathtt{Suc}(5_N)$

```
DEFINITION FS.4.13.7: 0-ary function 7_{N}. 7_{N} \simeq Suc(6_{N}).
```

DEFINITION FS.4.13.7: 0-ary function 7_N. $7_N \simeq \mathtt{Suc}(6_N)$.

$7_N \simeq \mathtt{Suc}(6_N)$

```
DEFINITION FS.4.13.8: 0-ary function 8_{N}. 8_{N} \simeq Suc(7_{N}).
```

DEFINITION FS.4.13.8: 0-ary function 8_N. $8_N \simeq \mathtt{Suc}(7_N)$.

$8_N \simeq \mathtt{Suc}(7_N)$

```
DEFINITION FS.4.13.9: 0-ary function 9_{N}. 9_{N} \simeq Suc(8_{N}).
```

DEFINITION FS.4.13.9: 0-ary function 9_N. $9_N \simeq \mathtt{Suc}(8_N)$.

$9_N \simeq \mathtt{Suc}(8_N)$

```
DEFINITION FS.4.13.10: 0-ary function 10_{N}. 10_{N} \simeq Suc(9_{N}).
```

DEFINITION FS.4.13.10: 0-ary function 10_N. $10_N \simeq \mathtt{Suc}(9_N)$.

$10_N \simeq \mathtt{Suc}(9_N)$

```
DEFINITION FS.4.14: 0-ary function Addnat. Addnat \simeq (!x)(
  (\forall y,z \in \omega)(
    x(y,0_{N}) = y \wedge x(y,Suc(z)) = Suc(x(y,z))
  ) \wedge
  (\forall y,z)(
    x(y,z)\dn \iff y,z \in \omega
  )
).
```

DEFINITION FS.4.14: 0-ary function Addnat. $\texttt{Addnat} \simeq (!x)((\forall y, z \in \omega)(x(y, \texttt{0}_\texttt{N}) = y \wedge x(y, \texttt{Suc}(z)) = \texttt{Suc}(x(y, z))) \wedge (\forall y, z)(x(y, z)\downarrow \leftrightarrow y, z \in \omega))$.

$\texttt{Addnat} \simeq (\iota y_0)(\exists x)(y_0 = x \wedge ((\forall y, z)(y \in \omega \wedge z \in \omega \rightarrow (\iota x_0)(\varpi_\texttt{0}(\varpi_\texttt{0}(y, \texttt{0}_\texttt{N}), x_0) \in x) = y \wedge (\iota x_0)(\varpi_\texttt{0}(\varpi_\texttt{0}(y, \texttt{Suc}(z)), x_0) \in x) = \texttt{Suc}((\iota x_0)(\varpi_\texttt{0}(\varpi_\texttt{0}(y, z), x_0) \in x))) \wedge (\forall y, z)(\iota x_0)(\varpi_\texttt{0}(\varpi_\texttt{0}(y, z), x_0) \in x)\downarrow \leftrightarrow y \in \omega \wedge z \in \omega))$

```
DEFINITION FS.4.15: Infix function +_{N}. x +_{N} y \simeq
(!z)(<x,y,z> \in Addnat). Precedence 60.
```

DEFINITION FS.4.15: Infix function $+_\texttt{N}$. $x +_\texttt{N} y \simeq (!z)(\langle x, y, z\rangle \in \texttt{Addnat})$. Precedence 60.

$+_\texttt{N}(x, y) \simeq (\iota x_0)(\exists z)(x_0 = z \wedge (\varpi_\texttt{0}(\varpi_\texttt{0}(x, y), z) \in \texttt{Addnat}))$

```
DEFINITION FS.4.16: 0-ary function Mulnat. Mulnat \simeq (!x)(
  (\forall y,z \in \omega)(
    x(y,0_{N}) = 0_{N} \wedge x(y,z +_{N} 1_{N}) = x(y,z) +_{N} y
  ) \wedge
  (\forall y,z)(
    x(y,z)\dn \iff y,z \in \omega
  )
).
```

DEFINITION FS.4.16: 0-ary function Mulnat. $\texttt{Mulnat} \simeq (!x)((\forall y, z \in \omega)(x(y, \texttt{0}_\texttt{N}) = \texttt{0}_\texttt{N} \wedge x(y, z +_N \texttt{1}_\texttt{N}) = x(y, z) +_N y) \wedge (\forall y, z)(x(y, z)\downarrow \leftrightarrow y, z \in \omega))$.

$\texttt{Mulnat} \simeq (\iota y_0)(\exists x)(y_0 = x \wedge ((\forall y, z)(y \in \omega \wedge z \in \omega \rightarrow (\iota x_0)(\varpi_\texttt{0}(\varpi_\texttt{0}(y, \texttt{0}_\texttt{N}), x_0) \in x) = \texttt{0}_\texttt{N} \wedge (\iota x_0)(\varpi_\texttt{0}(\varpi_\texttt{0}(y, +_\texttt{N}(z, \texttt{1}_\texttt{N})), x_0) \in x) = +_\texttt{N}((\iota x_0)(\varpi_\texttt{0}(\varpi_\texttt{0}(y, z), x_0) \in x), y)) \wedge (\forall y, z)(\iota x_0)(\varpi_\texttt{0}(\varpi_\texttt{0}(y, z), x_0) \in x)\downarrow \leftrightarrow y \in \omega \wedge z \in \omega))$

```
DEFINITION FS.4.17: Infix function \cdot_{N}. x \cdot_{N} y \simeq
(!z)(<x,y,z> \in Mulnat). Precedence 40.
```

DEFINITION FS.4.17: Infix function $\cdot_{\mathtt{N}}$. $x \cdot_{\mathtt{N}} y \simeq (!z)(\langle x, y, z\rangle \in \mathtt{Mulnat})$. Precedence 40.

$\cdot_{\mathtt{N}}(x, y) \simeq (\iota x_0)(\exists z)(x_0 = z \wedge (\varpi_{\mathtt{0}}(\varpi_{\mathtt{0}}(x, y), z) \in \mathtt{Mulnat}))$

```
DEFINITION FS.4.18: 0-ary function Expnat. Expnat \simeq (!x)(
  (\forall y,z \in \omega)(
    x(y,0_{N}) = 1_{N} \wedge x(y,z +_{N} 1_{N}) = x(y,z) \cdot_{N} y
  ) \wedge
  (\forall y,z)(
    x(y,z)\dn \iff y,z \in \omega
  )
).
```

DEFINITION FS.4.18: 0-ary function `Expnat`. $\mathtt{Expnat} \simeq (!x)((\forall y, z \in \omega)(x(y, \mathtt{0_N}) = \mathtt{1_N} \wedge x(y, z +_N \mathtt{1_N}) = x(y, z) \cdot_N y) \wedge (\forall y, z)(x(y, z)\downarrow \leftrightarrow y, z \in \omega))$.

$\mathtt{Expnat} \simeq (\iota y_0)(\exists x)(y_0 = x \wedge ((\forall y, z)(y \in \omega \wedge z \in \omega \rightarrow (\iota x_0)(\varpi_{\mathtt{0}}(\varpi_{\mathtt{0}}(y, \mathtt{0_N}), x_0) \in x) = \mathtt{1_N} \wedge (\iota x_0)(\varpi_{\mathtt{0}}(\varpi_{\mathtt{0}}(y, +_{\mathtt{N}}(z, \mathtt{1_N})), x_0) \in x) = \cdot_{\mathtt{N}}((\iota x_0)(\varpi_{\mathtt{0}}(\varpi_{\mathtt{0}}(y, z), x_0) \in x), y)) \wedge (\forall y, z)(\iota x_0)(\varpi_{\mathtt{0}}(\varpi_{\mathtt{0}}(y, z), x_0) \in x)\downarrow \leftrightarrow y \in \omega \wedge z \in \omega))$

```
DEFINITION FS.4.19: Infix function Exp_{N}. x Exp_{N} y \simeq
(!z)(<x,y,z> \in Expnat). Precedence 20.
```

DEFINITION FS.4.19: Infix function $\mathtt{Exp_N}$. $x \,\mathtt{Exp_N}\, y \simeq (!z)(\langle x, y, z\rangle \in \mathtt{Expnat})$. Precedence 20.

$\mathtt{Exp_N}(x, y) \simeq (\iota x_0)(\exists z)(x_0 = z \wedge (\varpi_{\mathtt{0}}(\varpi_{\mathtt{0}}(x, y), z) \in \mathtt{Expnat}))$

```
DEFINITION FS.4.20: 1-ary relation INF. INF[x] \iff \neg FIN[x].
```

DEFINITION FS.4.20: 1-ary relation `INF`. $\mathtt{INF}[x] \leftrightarrow \neg\, \mathtt{FIN}[x]$.

$\mathtt{INF}[x] \leftrightarrow \neg(\mathtt{FIN}[x])$

```
DEFINITION FS.4.21: 1-ary relation DEN. DEN[x] \iff
x \approx_{C} \omega.
```

DEFINITION FS.4.21: 1-ary relation DEN. $\texttt{DEN}[x] \leftrightarrow x \approx_{\texttt{C}} \omega$.

$\texttt{DEN}[x] \leftrightarrow \approx_{\texttt{C}}[x, \omega]$

```
DEFINITION FS.4.22: 1-ary relation DINF. DINF[x] \iff \neg DFIN[x].
```

DEFINITION FS.4.22: 1-ary relation DINF. $\texttt{DINF}[x] \leftrightarrow \neg\, \texttt{DFIN}[x]$.

$\texttt{DINF}[x] \leftrightarrow \neg(\texttt{DFIN}[x])$

```
DEFINITION FS.4.23: 1-ary relation CNTBL. CNTBL[A] \iff
(\exists f)(BIJ[f,\omega,A]).
```

DEFINITION FS.4.23: 1-ary relation CNTBL. $\texttt{CNTBL}[A] \leftrightarrow (\exists f)(\texttt{BIJ}[f, \omega, A])$.

$\texttt{CNTBL}[A] \leftrightarrow (\exists f)\, \texttt{BIJ}[f, \omega, A]$

```
DEFINITION FS.4.24: 1-ary relation UNCNTBL. UNCNTBL[A] \iff \neg CNTBL[A].
```

DEFINITION FS.4.24: 1-ary relation UNCNTBL. $\texttt{UNCNTBL}[A] \leftrightarrow \neg\, \texttt{CNTBL}[A]$.

$\texttt{UNCNTBL}[A] \leftrightarrow \neg(\texttt{CNTBL}[A])$

F.5 Chapter 6

```
DEFINITION FS.5.1: Infix function /. If x,y \in \omega \wedge y
\neq 0_{N} then x/y \simeq <x,y>. Otherwise x/y \up. Precedence 5.
```

DEFINITION FS.5.1: Infix function /. If $x, y \in \omega \wedge y \neq \mathtt{0_N}$ then $x \,/\, y \simeq \langle x, y \rangle$. Otherwise $x \,/\, y\uparrow$. Precedence 5.

$/(x, y) \simeq (\iota x_0)((x \in \omega \wedge y \in \omega \wedge \neg(y = \mathtt{0_N}) \wedge x_0 \simeq \varpi_{\mathtt{0}}(x, y)))$

```
DEFINITION FS.5.2: 0-ary function Fr. Fr \simeq {x/y : x/y\dn}.
```

DEFINITION FS.5.2: 0-ary function Fr. $\mathtt{Fr} \simeq \{x/y : x/y\downarrow\}$.

$\mathtt{Fr} \simeq (\iota y_0)(\forall x_0)(x_0 \in y_0 \leftrightarrow (\exists x, y)(x_0 = /(x, y) \wedge (/(x, y)\downarrow)))$

```
DEFINITION FS.5.3: Infix relation \equiv_{Fr}. x \equiv_{Fr} y \iff
(\exists a,b,c,d)(
  x = a/b \wedge y = c/d \wedge a \cdot_{N} d = b \cdot_{N} c
).
```

DEFINITION FS.5.3: Infix relation $\equiv_{\mathtt{Fr}}$. $x \equiv_{\mathtt{Fr}} y \leftrightarrow (\exists a, b, c, d)(x = a/b \wedge y = c/d \wedge a \cdot_N d = b \cdot_N c)$.

$\equiv_{\mathtt{Fr}}[x, y] \leftrightarrow (\exists a, b, c, d)x = /(a, b) \wedge y = /(c, d) \wedge \cdot_{\mathtt{N}}(a, d) = \cdot_{\mathtt{N}}(b, c)$

```
DEFINITION FS.5.4: Infix relation <_{Fr}. x <_{Fr} y \iff
(\exists a,b,c,d)(
  x = a/b \wedge y = c/d \wedge a \cdot_{N} d <_{O} b \cdot_{N} c
).
```

DEFINITION FS.5.4: Infix relation $<_{\mathtt{Fr}}$. $x <_{\mathtt{Fr}} y \leftrightarrow (\exists a, b, c, d)(x = a/b \wedge y = c/d \wedge a \cdot_N d <_{\mathtt{0}} b \cdot_N c)$.

$<_{\mathtt{Fr}}[x, y] \leftrightarrow (\exists a, b, c, d)x = /(a, b) \wedge y = /(c, d) \wedge <_{\mathtt{0}}[\cdot_{\mathtt{N}}(a, d), \cdot_{\mathtt{N}}(b, c)]$

```
DEFINITION FS.5.5: Infix relation >_{Fr}. x >_{Fr} y \iff y <_{Fr} x.
```

DEFINITION FS.5.5: Infix relation $>_{\mathtt{Fr}}$. $x >_{\mathtt{Fr}} y \leftrightarrow y <_{\mathtt{Fr}} x$.

$>_{\mathtt{Fr}}[x, y] \leftrightarrow <_{\mathtt{Fr}}[y, x]$

```
DEFINITION FS.5.6: Infix relation \leq_{Fr}. x \leq_{Fr} y \iff
x <_{Fr} y \vee x \equiv_{Fr} y.
```

DEFINITION FS.5.6: Infix relation $\leq_{\mathtt{Fr}}$. $x \leq_{\mathtt{Fr}} y \leftrightarrow x <_{\mathtt{Fr}} y \vee x \equiv_{\mathtt{Fr}} y$.

$\leq_{\mathtt{Fr}}[x, y] \leftrightarrow <_{\mathtt{Fr}}[x, y] \vee \equiv_{\mathtt{Fr}}[x, y]$

```
DEFINITION FS.5.7: Infix relation \geq_{Fr}. x \geq_{Fr} y \iff
x >_{Fr} y \vee x \equiv_{Fr} y.
```

DEFINITION FS.5.7: Infix relation $\geq_{\mathtt{Fr}}$. $x \geq_{\mathtt{Fr}} y \leftrightarrow x >_{\mathtt{Fr}} y \vee x \equiv_{\mathtt{Fr}} y$.

$\geq_{\mathtt{Fr}}[x, y] \leftrightarrow >_{\mathtt{Fr}}[x, y] \vee \equiv_{\mathtt{Fr}}[x, y]$

```
DEFINITION FS.5.8: Infix function +_{Fr}. x +_{Fr} y \simeq
(!z)(\exists a,b,c,d,e,f)(
  x = a/b \wedge y = c/d \wedge z = e/f \wedge
  e = a \cdot_{N} d +_{N} b \cdot_{N} c
  \wedge f = b \cdot_{N} d
). Precedence 40.
```

DEFINITION FS.5.8: Infix function $+_{\mathtt{Fr}}$. $x +_{\mathtt{Fr}} y \simeq (!z)(\exists a, b, c, d, e, f)(x = a/b \wedge y = c/d \wedge z = e/f \wedge e = a \cdot_N d +_N b \cdot_N c \wedge f = b \cdot_N d)$. Precedence 40.

$+_{\mathtt{Fr}}(x, y) \simeq (\iota x_0)(\exists z)(x_0 = z \wedge ((\exists a, b, c, d, e, f)x = /(a, b) \wedge y = /(c, d) \wedge z = /(e, f) \wedge e = \cdot_{\mathtt{N}}(+_{\mathtt{N}}(\cdot_{\mathtt{N}}(a, d), b), c) \wedge f = \cdot_{\mathtt{N}}(b, d)))$

```
DEFINITION FS.5.9: Infix function \cdot_{Fr}. x \cdot_{Fr} y \simeq (!t)
(\exists a,b,c,d,e,f)(
  x = a/b \wedge y = c/d \wedge t = e/f \wedge e = a \cdot_{N} c \wedge
  f = b \cdot_{N} d
). Precedence 20.
```

DEFINITION FS.5.9: Infix function $\cdot_{\mathtt{Fr}}$. $x \cdot_{\mathtt{Fr}} y \simeq (!t)(\exists a, b, c, d, e, f)(x = a/b \wedge y = c/d \wedge t = e/f \wedge e = a \cdot_N c \wedge f = b \cdot_N d)$. Precedence 20.

$\cdot_{\mathtt{Fr}}(x, y) \simeq (\iota x_0)(\exists t)(x_0 = t \wedge ((\exists a, b, c, d, e, f)x = /(a, b) \wedge y = /(c, d) \wedge t = /(e, f) \wedge e = \cdot_{\mathtt{N}}(a, c) \wedge f = \cdot_{\mathtt{N}}(b, d)))$

```
DEFINITION FS.5.10: 0-ary function Nra. Nra \simeq
{Coset(x,{<u,v> : u \equiv_{Fr} v}) : x \in Fr}.
```

DEFINITION FS.5.10: 0-ary function `Nra`. $\mathtt{Nra} \simeq \{\mathtt{Coset}(x, \{\langle u, v\rangle : u \equiv_{\mathtt{Fr}} v\}) : x \in \mathtt{Fr}\}$.

$\mathtt{Nra} \simeq (\iota x_1)(\forall z_0)(z_0 \in x_1 \leftrightarrow (\exists x)(z_0 = \mathtt{Coset}(x, (\iota y_0)(\forall x_0)(x_0 \in y_0 \leftrightarrow (\exists u, v)(x_0 = \varpi_0(u, v) \wedge (\equiv_{\mathtt{Fr}}[u, v])))) \wedge (x \in \mathtt{Fr})))$

```
DEFINITION FS.5.10.5: 1-ary function Incl_{FrNra}.
If x \in Fr then Incl_{FrNra}(x) \simeq
Coset(x,{<u,v> : u \equiv_{Fr} v}). Otherwise Incl_{FrNra}(x) \up.
```

DEFINITION FS.5.10.5: 1-ary function $\mathtt{Incl}_{\mathtt{FrNra}}$. If $x \in \mathtt{Fr}$ then $\mathtt{Incl}_{\mathtt{FrNra}}(x) \simeq \mathtt{Coset}(x, \{\langle u, v\rangle : u \equiv_{\mathtt{Fr}} v\})$. Otherwise $\mathtt{Incl}_{\mathtt{FrNra}}(x)\uparrow$.

$\mathtt{Incl}_{\mathtt{FrNra}}(x) \simeq (\iota z_0)((x \in \mathtt{Fr} \wedge z_0 \simeq \mathtt{Coset}(x, (\iota y_0)(\forall x_0)(x_0 \in y_0 \leftrightarrow (\exists u, v)(x_0 = \varpi_0(u, v) \wedge (\equiv_{\mathtt{Fr}}[u, v])))))$

```
DEFINITION FS.5.11: Infix relation <_{Nra}. x <_{Nra} y \iff
x,y \in Nra \wedge (\exists u,v)(
  u \in x \wedge v \in y \wedge u <_{Fr} v
).
```

DEFINITION FS.5.11: Infix relation $<_{\texttt{Nra}}$. $x <_{\texttt{Nra}} y \leftrightarrow x, y \in \texttt{Nra} \wedge (\exists u, v)(u \in x \wedge v \in y \wedge u <_{\texttt{Fr}} v)$.

$<_{\texttt{Nra}}[x, y] \leftrightarrow x \in \texttt{Nra} \wedge y \in \texttt{Nra} \wedge (\exists u, v) u \in x \wedge v \in y \wedge <_{\texttt{Fr}}[u, v]$

```
DEFINITION FS.5.12: Infix relation >_{Nra}. x >_{Nra} y \iff
x,y \in Nra \wedge (\exists u,v)(
  u \in x \wedge v \in y \wedge u >_{Fr} v
).
```

DEFINITION FS.5.12: Infix relation $>_{\texttt{Nra}}$. $x >_{\texttt{Nra}} y \leftrightarrow x, y \in \texttt{Nra} \wedge (\exists u, v)(u \in x \wedge v \in y \wedge u >_{\texttt{Fr}} v)$.

$>_{\texttt{Nra}}[x, y] \leftrightarrow x \in \texttt{Nra} \wedge y \in \texttt{Nra} \wedge (\exists u, v) u \in x \wedge v \in y \wedge >_{\texttt{Fr}}[u, v]$

```
DEFINITION FS.5.13: Infix relation \leq_{Nra}. x \leq_{Nra} y \iff
x,y \in Nra \wedge
(\exists u,v)(u \in x \wedge v \in y \wedge u \leq_{Fr} v).
```

DEFINITION FS.5.13: Infix relation $\leq_{\texttt{Nra}}$. $x \leq_{\texttt{Nra}} y \leftrightarrow x, y \in \texttt{Nra} \wedge (\exists u, v)(u \in x \wedge v \in y \wedge u \leq_{\texttt{Fr}} v)$.

$\leq_{\texttt{Nra}}[x, y] \leftrightarrow x \in \texttt{Nra} \wedge y \in \texttt{Nra} \wedge (\exists u, v) u \in x \wedge v \in y \wedge \leq_{\texttt{Fr}}[u, v]$

```
DEFINITION FS.5.14: Infix relation \geq_{Nra}. x \geq_{Nra} y \iff
x,y \in Nra \wedge
(\exists u,v)(u \in x \wedge v \in y \wedge u \geq_{Fr} v).
```

DEFINITION FS.5.14: Infix relation $\geq_{\texttt{Nra}}$. $x \geq_{\texttt{Nra}} y \leftrightarrow x, y \in \texttt{Nra} \wedge (\exists u, v)(u \in x \wedge v \in y \wedge u \geq_{\texttt{Fr}} v)$.

$\geq_{\texttt{Nra}}[x, y] \leftrightarrow x \in \texttt{Nra} \wedge y \in \texttt{Nra} \wedge (\exists u, v) u \in x \wedge v \in y \wedge \geq_{\texttt{Fr}}[u, v]$

```
DEFINITION FS.5.15: Infix function +_{Nra}. x +_{Nra} y \simeq (!z)(
  x,y,z \in Nra \wedge (\exists u,v,w)(
    u \in x \wedge v \in y \wedge w \in z \wedge
    u +_{Fr} v \equiv_{Fr} w
  )
). Precedence 40.
```

DEFINITION FS.5.15: Infix function $+_{\mathtt{Nra}}$. $x +_{\mathtt{Nra}} y \simeq (!z)(x, y, z \in \mathtt{Nra} \wedge (\exists u, v, w)(u \in x \wedge v \in y \wedge w \in z \wedge u +_{Fr} v \equiv_{\mathtt{Fr}} w))$. Precedence 40.

$+_{\mathtt{Nra}}(x, y) \simeq (\iota x_0)(\exists z)(x_0 = z \wedge (x \in \mathtt{Nra} \wedge y \in \mathtt{Nra} \wedge z \in \mathtt{Nra} \wedge (\exists u, v, w) u \in x \wedge v \in y \wedge w \in z \wedge \equiv_{\mathtt{Fr}}[+_{\mathtt{Fr}}(u, v), w]))$

```
DEFINITION FS.5.16: Infix function \cdot_{Nra}. x \cdot_{Nra} y \simeq
(!z)(
  x,y,z \in Nra \wedge (\exists u,v,w)(
    u \in x \wedge v \in y \wedge w \in z \wedge
    u \cdot_{Fr} v \equiv_{Fr} w
  )
). Precedence 20.
```

DEFINITION FS.5.16: Infix function $\cdot_{\mathtt{Nra}}$. $x \cdot_{\mathtt{Nra}} y \simeq (!z)(x, y, z \in \mathtt{Nra} \wedge (\exists u, v, w)(u \in x \wedge v \in y \wedge w \in z \wedge u \cdot_{Fr} v \equiv_{\mathtt{Fr}} w))$. Precedence 20.

$\cdot_{\mathtt{Nra}}(x, y) \simeq (\iota x_0)(\exists z)(x_0 = z \wedge (x \in \mathtt{Nra} \wedge y \in \mathtt{Nra} \wedge z \in \mathtt{Nra} \wedge (\exists u, v, w) u \in x \wedge v \in y \wedge w \in z \wedge \equiv_{\mathtt{Fr}}[\cdot_{\mathtt{Fr}}(u, v), w]))$

```
DEFINITION FS.5.17: 0-ary function 0_{Nra}. 0_{Nra} \simeq Coset(
  0_{N}/1_{N},{<x,y> : x \equiv_{Fr} y}
).
```

DEFINITION FS.5.17: 0-ary function $\mathtt{0}_{\mathtt{Nra}}$. $\mathtt{0}_{\mathtt{Nra}} \simeq \mathtt{Coset}(\mathtt{0}_{\mathtt{N}} \,/\, \mathtt{1}_{\mathtt{N}}, \{\langle x, y\rangle : x \equiv_{\mathtt{Fr}} y\})$.

$\mathtt{0}_{\mathtt{Nra}} \simeq \mathtt{Coset}(/(\mathtt{0}_{\mathtt{N}}, \mathtt{1}_{\mathtt{N}}), (\iota y_0)(\forall x_0)(x_0 \in y_0 \leftrightarrow (\exists x, y)(x_0 = \varpi_0(x, y) \wedge (\equiv_{\mathtt{Fr}}[x, y]))))$

```
DEFINITION FS.5.18: 0-ary function 1_{Nra}. 1_{Nra} \simeq Coset(
  1_{N}/1_{N},{<x,y> : x \equiv_{Fr} y}
).
```

DEFINITION FS.5.18: 0-ary function $1_{\mathtt{Nra}}$. $1_{\mathtt{Nra}} \simeq \mathtt{Coset}(1_{\mathtt{N}} \,/\, 1_{\mathtt{N}}, \{\langle x, y\rangle : x \equiv_{\mathtt{Fr}} y\})$.

$1_{\mathtt{Nra}} \simeq \mathtt{Coset}(/(1_{\mathtt{N}}, 1_{\mathtt{N}}), (\iota y_0)(\forall x_0)(x_0 \in y_0 \leftrightarrow (\exists x, y)(x_0 = \varpi_0(x, y) \wedge (\equiv_{\mathtt{Fr}}[x, y]))))$

```
DEFINITION FS.5.19: Infix relation \equiv_{SUB}. x \equiv_{SUB} y \iff
(\exists a,b,c,d)(
  x = <a,b> \wedge y = <c,d> \wedge a +_{Nra} d = b +_{Nra} c
).
```

DEFINITION FS.5.19: Infix relation $\equiv_{\mathtt{SUB}}$. $x \equiv_{\mathtt{SUB}} y \leftrightarrow (\exists a, b, c, d)(x = \langle a, b\rangle \wedge y = \langle c, d\rangle \wedge a +_{Nra} d = b +_{Nra} c)$.

$\equiv_{\mathtt{SUB}}[x, y] \leftrightarrow (\exists a, b, c, d)x = \varpi_0(a, b) \wedge y = \varpi_0(c, d) \wedge +_{\mathtt{Nra}}(a, d) = +_{\mathtt{Nra}}(b, c)$

```
DEFINITION FS.5.20: Infix relation <_{SUB}. x <_{SUB} y \iff
(\exists a,b,c,d)(
  x = <a,b> \wedge y = <c,d> \wedge a +_{Nra} d <_{Nra} b +_{Nra} c
).
```

DEFINITION FS.5.20: Infix relation $<_{\mathtt{SUB}}$. $x <_{\mathtt{SUB}} y \leftrightarrow (\exists a, b, c, d)(x = \langle a, b\rangle \wedge y = \langle c, d\rangle \wedge a +_{Nra} d <_{\mathtt{Nra}} b +_{Nra} c)$.

$<_{\mathtt{SUB}}[x, y] \leftrightarrow (\exists a, b, c, d)x = \varpi_0(a, b) \wedge y = \varpi_0(c, d) \wedge <_{\mathtt{Nra}}[+_{\mathtt{Nra}}(a, d), +_{\mathtt{Nra}}(b, c)]$

```
DEFINITION FS.5.21: Infix function +_{SUB}. x +_{SUB} y \simeq (!z)
(\exists a,b,c,d,e,f)(
  x = <a,b> \wedge y = <c,d> \wedge z = <e,f> \wedge
  a +_{Nra} c +_{Nra} f = b +_{Nra} d +_{Nra} e
). Precedence 40.
```

DEFINITION FS.5.21: Infix function $+_{\mathtt{SUB}}$. $x +_{\mathtt{SUB}} y \simeq (!z)(\exists a, b, c, d, e, f)(x = \langle a, b\rangle \wedge y = \langle c, d\rangle \wedge z = \langle e, f\rangle \wedge a +_{Nra} c +_{Nra} f = b +_{Nra} d +_{Nra} e)$. Precedence 40.

$+_{\mathtt{SUB}}(x, y) \simeq (\iota x_0)(\exists z)(x_0 = z \wedge ((\exists a, b, c, d, e, f)x = \varpi_{\mathtt{0}}(a, b) \wedge y = \varpi_{\mathtt{0}}(c, d) \wedge z = \varpi_{\mathtt{0}}(e, f) \wedge +_{\mathtt{Nra}}(+_{\mathtt{Nra}}(a, c), f) = +_{\mathtt{Nra}}(+_{\mathtt{Nra}}(b, d), e)))$

```
DEFINITION FS.5.22: Infix function \cdot_{SUB}.
x \cdot_{SUB} y \simeq
(!z)(\exists a,b,c,d,e,f)(
  x = <a,b> \wedge y = <c,d> \wedge z = <e,f> \wedge
  (a \cdot_{Nra} c) +_{Nra} (b \cdot_{Nra} d) +_{Nra} f =
    (a \cdot_{Nra} d) +_{Nra} (b \cdot_{Nra} c) +_{Nra} e
). Precedence 20.
```

DEFINITION FS.5.22: Infix function $\cdot_{\mathtt{SUB}}$. $x \cdot_{\mathtt{SUB}} y \simeq (!z)(\exists a, b, c, d, e, f)(x = \langle a, b\rangle \wedge y = \langle c, d\rangle \wedge z = \langle e, f\rangle \wedge (a \cdot_{Nra} c) +_{Nra} (b \cdot_{Nra} d) +_{Nra} f = (a \cdot_{Nra} d) +_{Nra} (b \cdot_{Nra} c) +_{Nra} e)$. Precedence 20.

$\cdot_{\mathtt{SUB}}(x, y) \simeq (\iota x_0)(\exists z)(x_0 = z \wedge ((\exists a, b, c, d, e, f)x = \varpi_{\mathtt{0}}(a, b) \wedge y = \varpi_{\mathtt{0}}(c, d) \wedge z = \varpi_{\mathtt{0}}(e, f) \wedge +_{\mathtt{Nra}}(+_{\mathtt{Nra}}((\cdot_{\mathtt{Nra}}(a, c)), (\cdot_{\mathtt{Nra}}(b, d))), f) = +_{\mathtt{Nra}}(+_{\mathtt{Nra}}((\cdot_{\mathtt{Nra}}(a, d)), (\cdot_{\mathtt{Nra}}(b, c))), e)))$

```
DEFINITION FS.5.23: 0-ary function Ra. Ra \simeq
{Coset(x,{<u,v> : u \equiv_{SUB} v}) : x \in (Nra \times Nra)}.
```

DEFINITION FS.5.23: 0-ary function Ra. $\mathtt{Ra} \simeq \{\mathtt{Coset}(x, \{\langle u, v\rangle : u \equiv_{\mathtt{SUB}} v\}) : x \in (\mathtt{Nra} \times \mathtt{Nra})\}$.

$\mathtt{Ra} \simeq (\iota x_1)(\forall z_0)(z_0 \in x_1 \leftrightarrow (\exists x)(z_0 = \mathtt{Coset}(x, (\iota y_0)(\forall x_0)(x_0 \in y_0 \leftrightarrow (\exists u, v)(x_0 = \varpi_{\mathtt{0}}(u, v) \wedge (\equiv_{\mathtt{SUB}}[u, v])))) \wedge (x \in (\times(\mathtt{Nra}, \mathtt{Nra})))))$

```
DEFINITION FS.5.23.5: 1-ary function Incl_{NraRa}. If x \in Nra then
Incl_{NraRa}(x) \simeq Coset(<x,0_{Nra}>,{<u,v> : u \equiv_{SUB} v}).
```

DEFINITION FS.5.23.5: 1-ary function $\texttt{Incl}_{\texttt{NraRa}}$. If $x \in \texttt{Nra}$ then $\texttt{Incl}_{\texttt{NraRa}}(x) \simeq \texttt{Coset}(\langle x, \texttt{0}_{\texttt{Nra}} \rangle, \{\langle u, v \rangle : u \equiv_{\texttt{SUB}} v\})$.

$\texttt{Incl}_{\texttt{NraRa}}(x) \simeq (\iota z_0)(x \in \texttt{Nra} \wedge z_0 \simeq \texttt{Coset}(\varpi_0(x, \texttt{0}_{\texttt{Nra}}), (\iota y_0)(\forall x_0)(x_0 \in y_0 \leftrightarrow (\exists u, v)(x_0 = \varpi_0(u, v) \wedge (\equiv_{\texttt{SUB}}[u, v])))))$

```
DEFINITION FS.5.23.8: 1-ary function Incl_{FrRa}.
If x \in Fr then Incl_{FrRa}(x) \simeq Incl_{NraRa}(Incl_{FrNra}(x)).
```

DEFINITION FS.5.23.8: 1-ary function $\texttt{Incl}_{\texttt{FrRa}}$. If $x \in \texttt{Fr}$ then $\texttt{Incl}_{\texttt{FrRa}}(x) \simeq \texttt{Incl}_{\texttt{NraRa}}(\texttt{Incl}_{\texttt{FrNra}}(x))$.

$\texttt{Incl}_{\texttt{FrRa}}(x) \simeq (\iota x_0)(x \in \texttt{Fr} \wedge x_0 \simeq \texttt{Incl}_{\texttt{NraRa}}(\texttt{Incl}_{\texttt{FrNra}}(x)))$

```
DEFINITION FS.5.23.A: 0-ary function \mathbb{Q}. \mathbb{Q} \simeq Ra.
```

DEFINITION FS.5.23.A: 0-ary function $\mathbb{Q}$. $\mathbb{Q} \simeq \texttt{Ra}$.

$\mathbb{Q} \simeq \texttt{Ra}$

```
DEFINITION FS.5.24: Infix relation <_{Ra}. x <_{Ra} y \iff
(\exists z,w)(
  x,y \in Ra \wedge z \in x \wedge w \in y \wedge z <_{SUB} w
).
```

DEFINITION FS.5.24: Infix relation $<_{\texttt{Ra}}$. $x <_{\texttt{Ra}} y \leftrightarrow (\exists z, w)(x, y \in \texttt{Ra} \wedge z \in x \wedge w \in y \wedge z <_{\texttt{SUB}} w)$.

$<_{\texttt{Ra}}[x, y] \leftrightarrow (\exists z, w) x \in \texttt{Ra} \wedge y \in \texttt{Ra} \wedge z \in x \wedge w \in y \wedge <_{\texttt{SUB}}[z, w]$

```
DEFINITION FS.5.25: Infix function +_{Ra}. x +_{Ra} y \simeq (!z)(
  x,y,z \in Ra \wedge (\exists a,b,c)(
    a \in x \wedge b \in y \wedge c \in z \wedge a +_{SUB} b = c
  )
). Precedence 40.
```

DEFINITION FS.5.25: Infix function $+_{\mathtt{Ra}}$. $x +_{\mathtt{Ra}} y \simeq (!z)(x, y, z \in \mathtt{Ra} \wedge (\exists a, b, c)(a \in x \wedge b \in y \wedge c \in z \wedge a +_{SUB} b = c))$. Precedence 40.

$+_{\mathtt{Ra}}(x, y) \simeq (\iota x_0)(\exists z)(x_0 = z \wedge (x \in \mathtt{Ra} \wedge y \in \mathtt{Ra} \wedge z \in \mathtt{Ra} \wedge (\exists a, b, c) a \in x \wedge b \in y \wedge c \in z \wedge +_{\mathtt{SUB}}(a, b) = c))$

```
DEFINITION FS.5.26: Infix function \cdot_{Ra}. x \cdot_{Ra} y \simeq
(!z)(
  x,y,z \in Ra \wedge (\exists a,b,c)(
    a \in x \wedge b \in y \wedge c \in z \wedge a \cdot_{SUB} b = c
  )
). Precedence 20.
```

DEFINITION FS.5.26: Infix function $\cdot_{\mathtt{Ra}}$. $x \cdot_{\mathtt{Ra}} y \simeq (!z)(x, y, z \in \mathtt{Ra} \wedge (\exists a, b, c)(a \in x \wedge b \in y \wedge c \in z \wedge a \cdot_{SUB} b = c))$. Precedence 20.

$\cdot_{\mathtt{Ra}}(x, y) \simeq (\iota x_0)(\exists z)(x_0 = z \wedge (x \in \mathtt{Ra} \wedge y \in \mathtt{Ra} \wedge z \in \mathtt{Ra} \wedge (\exists a, b, c) a \in x \wedge b \in y \wedge c \in z \wedge \cdot_{\mathtt{SUB}}(a, b) = c))$

```
DEFINITION FS.5.27: 0-ary function 0_{Ra}. 0_{Ra} \simeq Coset(
  <0_{Nra},0_{Nra}>, {<x,y> : x \equiv_{SUB} y}
).
```

DEFINITION FS.5.27: 0-ary function $\mathtt{0}_{\mathtt{Ra}}$. $\mathtt{0}_{\mathtt{Ra}} \simeq \mathtt{Coset}(\langle \mathtt{0}_{\mathtt{Nra}}, \mathtt{0}_{\mathtt{Nra}} \rangle, \{\langle x, y \rangle : x \equiv_{\mathtt{SUB}} y\})$.

$\mathtt{0}_{\mathtt{Ra}} \simeq \mathtt{Coset}(\varpi_0(\mathtt{0}_{\mathtt{Nra}}, \mathtt{0}_{\mathtt{Nra}}), (\iota y_0)(\forall x_0)(x_0 \in y_0 \leftrightarrow (\exists x, y)(x_0 = \varpi_0(x, y) \wedge (\equiv_{\mathtt{SUB}}[x, y]))))$

```
DEFINITION FS.5.28: 0-ary function 1_{Ra}. 1_{Ra} \simeq Coset(
  <1_{Nra},0_{Nra}>, {<x,y> : x \equiv_{SUB} y}
).
```

DEFINITION FS.5.28: 0-ary function $\mathtt{1}_{\mathtt{Ra}}$. $\mathtt{1}_{\mathtt{Ra}} \simeq \mathtt{Coset}(\langle \mathtt{1}_{\mathtt{Nra}}, \mathtt{0}_{\mathtt{Nra}} \rangle, \{\langle x, y \rangle : x \equiv_{\mathtt{SUB}} y\})$.

$\mathtt{1}_{\mathtt{Ra}} \simeq \mathtt{Coset}(\varpi_0(\mathtt{1}_{\mathtt{Nra}}, \mathtt{0}_{\mathtt{Nra}}), (\iota y_0)(\forall x_0)(x_0 \in y_0 \leftrightarrow (\exists x, y)(x_0 = \varpi_0(x, y) \wedge (\equiv_{\mathtt{SUB}}[x, y]))))$

```
DEFINITION FS.5.29: Infix relation >_{Ra}. x >_{Ra} y \iff y <_{Ra} x.
```

DEFINITION FS.5.29: Infix relation $>_{\mathtt{Ra}}$. $x >_{\mathtt{Ra}} y \leftrightarrow y <_{\mathtt{Ra}} x$.

$>_{\mathtt{Ra}}[x, y] \leftrightarrow <_{\mathtt{Ra}}[y, x]$

```
DEFINITION FS.5.30: Infix relation \leq_{Ra}. x \leq_{Ra} y \iff
x <_{Ra} y \vee x = y.
```

DEFINITION FS.5.30: Infix relation $\leq_{\mathtt{Ra}}$. $x \leq_{\mathtt{Ra}} y \leftrightarrow x <_{\mathtt{Ra}} y \vee x = y$.

$\leq_{\mathtt{Ra}}[x, y] \leftrightarrow <_{\mathtt{Ra}}[x, y] \vee x = y$

```
DEFINITION FS.5.31: Infix relation \geq_{Ra}. x \geq_{Ra} y \iff
x >_{Ra} y \vee x = y.
```

DEFINITION FS.5.31: Infix relation $\geq_{\mathtt{Ra}}$. $x \geq_{\mathtt{Ra}} y \leftrightarrow x >_{\mathtt{Ra}} y \vee x = y$.

$\geq_{\mathtt{Ra}}[x, y] \leftrightarrow >_{\mathtt{Ra}}[x, y] \vee x = y$

```
DEFINITION FS.5.32: Infix function -_{Ra}. x -_{Ra} y \simeq
(!z)(x = y +_{Ra} z). Precedence 60.
```

DEFINITION FS.5.32: Infix function $-_{\mathtt{Ra}}$. $x -_{\mathtt{Ra}} y \simeq (!z)(x = y +_{Ra} z)$. Precedence 60.

$-_{\mathtt{Ra}}(x, y) \simeq (\iota x_0)(\exists z)(x_0 = z \wedge (x = +_{\mathtt{Ra}}(y, z)))$

```
DEFINITION FS.5.33: 1-ary function Av_{Ra}. Av_{Ra}(x) \simeq
(!y \in Ra)(
  (x \geq_{Ra} 0_{Ra} \implies y = x) \wedge
  (x <_{Ra} 0_{Ra} \implies y = 0_{Ra} -_{Ra} x)
).
```

DEFINITION FS.5.33: 1-ary function $\mathtt{Av}_{\mathtt{Ra}}$. $\mathtt{Av}_{\mathtt{Ra}}(x) \simeq (!y \in \mathtt{Ra})((x \geq_{\mathtt{Ra}} \mathtt{0}_{\mathtt{Ra}} \rightarrow y = x) \wedge (x <_{\mathtt{Ra}} \mathtt{0}_{\mathtt{Ra}} \rightarrow y = \mathtt{0}_{\mathtt{Ra}} -_{Ra} x))$.

$\mathtt{Av}_{\mathtt{Ra}}(x) \simeq (\iota x_0)(\exists y)(x_0 = y \wedge (\geq_{\mathtt{Ra}}[x, \mathtt{0}_{\mathtt{Ra}}] \rightarrow y = x \wedge <_{\mathtt{Ra}}[x, \mathtt{0}_{\mathtt{Ra}}] \rightarrow y = -_{\mathtt{Ra}}(\mathtt{0}_{\mathtt{Ra}}, x)) \wedge (y \in \mathtt{Ra}))$

```
DEFINITION FS.5.35: 0-ary function Nat_{Ra}. Nat_{Ra} \simeq
(!x)(\forall y)(
  y \in x \iff (
    y = 0_{Ra} \vee (
      y >_{Ra} 0_{Ra} \wedge y -_{Ra} 1_{Ra} \in x
    )
  )
).
```

DEFINITION FS.5.35: 0-ary function $\mathtt{Nat}_{\mathtt{Ra}}$. $\mathtt{Nat}_{\mathtt{Ra}} \simeq (!x)(\forall y)(y \in x \leftrightarrow (y = \mathtt{0}_{\mathtt{Ra}} \vee (y >_{\mathtt{Ra}} \mathtt{0}_{\mathtt{Ra}} \wedge y -_{Ra} \mathtt{1}_{\mathtt{Ra}} \in x)))$.

$\mathtt{Nat}_{\mathtt{Ra}} \simeq (\iota x_0)(\exists x)(x_0 = x \wedge ((\forall y) y \in x \leftrightarrow y = \mathtt{0}_{\mathtt{Ra}} \vee >_{\mathtt{Ra}}[y, \mathtt{0}_{\mathtt{Ra}}] \wedge -_{\mathtt{Ra}}(y, \mathtt{1}_{\mathtt{Ra}}) \in x))$

```
DEFINITION FS.5.35.A: 0-ary function \mathbb{N}_{Ra}.
\mathbb{N}_{Ra} \simeq Nat_{Ra}.
```

DEFINITION FS.5.35.A: 0-ary function $\mathbb{N}_{\mathtt{Ra}}$. $\mathbb{N}_{\mathtt{Ra}} \simeq \mathtt{Nat}_{\mathtt{Ra}}$.

$\mathbb{N}_{\mathtt{Ra}} \simeq \mathtt{Nat}_{\mathtt{Ra}}$

```
DEFINITION FS.5.36: 0-ary function Int_{Ra}. Int_{Ra} \simeq
(!x)(\forall y)(
  y \in x \iff (
    y \in Nat_{Ra} \vee 0_{Ra} -_{Ra} y \in Nat_{Ra}
  )
).
```

DEFINITION FS.5.36: 0-ary function $\texttt{Int}_{\texttt{Ra}}$. $\texttt{Int}_{\texttt{Ra}} \simeq (!x)(\forall y)(y \in x \leftrightarrow (y \in \texttt{Nat}_{\texttt{Ra}} \vee \texttt{0}_{\texttt{Ra}} -_{Ra} y \in \texttt{Nat}_{\texttt{Ra}}))$.

$\texttt{Int}_{\texttt{Ra}} \simeq (\iota x_0)(\exists x)(x_0 = x \wedge ((\forall y) y \in x \leftrightarrow y \in \texttt{Nat}_{\texttt{Ra}} \vee -_{\texttt{Ra}}(\texttt{0}_{\texttt{Ra}}, y) \in \texttt{Nat}_{\texttt{Ra}}))$

```
DEFINITION FS.5.36.A: 0-ary function \mathbb{Z}_{Ra}.
\mathbb{Z}_{Ra} \simeq Int_{Ra}.
```

DEFINITION FS.5.36.A: 0-ary function $\mathbb{Z}_{\texttt{Ra}}$. $\mathbb{Z}_{\texttt{Ra}} \simeq \texttt{Int}_{\texttt{Ra}}$.

$\mathbb{Z}_{\texttt{Ra}} \simeq \texttt{Int}_{\texttt{Ra}}$

```
DEFINITION FS.5.37: 0-ary function Seq_{Ra}.
Seq_{Ra} \simeq Maps(\omega,Ra).
```

DEFINITION FS.5.37: 0-ary function $\texttt{Seq}_{\texttt{Ra}}$. $\texttt{Seq}_{\texttt{Ra}} \simeq \texttt{Maps}(\omega, \texttt{Ra})$.

$\texttt{Seq}_{\texttt{Ra}} \simeq \texttt{Maps}(\omega, \texttt{Ra})$

```
DEFINITION FS.5.38: Infix function +_{SeqRa}.
x +_{SeqRa} y \simeq (!z)(
  x,y,z \in Seq_{Ra} \wedge
  (\forall n \in \omega)(z(n) = x(n) +_{Ra} y(n))
). Precedence 40.
```

DEFINITION FS.5.38: Infix function $+_{\texttt{SeqRa}}$. $x +_{\texttt{SeqRa}} y \simeq (!z)(x, y, z \in \texttt{Seq}_{\texttt{Ra}} \wedge (\forall n \in \omega)(z(n) = x(n) +_{Ra} y(n)))$. Precedence 40.

$+_{\texttt{SeqRa}}(x, y) \simeq (\iota y_0)(\exists z)(y_0 = z \wedge (x \in \texttt{Seq}_{\texttt{Ra}} \wedge y \in \texttt{Seq}_{\texttt{Ra}} \wedge z \in \texttt{Seq}_{\texttt{Ra}} \wedge (\forall n)(n \in \omega \rightarrow (\iota x_0)(\varpi_0(n, x_0) \in z) = +_{\texttt{Ra}}((\iota x_0)(\varpi_0(n, x_0) \in x), (\iota x_0)(\varpi_0(n, x_0) \in y)))))$

```
DEFINITION FS.5.39: Infix function \cdot_{SeqRa}.
x \cdot_{SeqRa} y \simeq (!z)(
  x,y,z \in Seq_{Ra} \wedge
  (\forall n \in \omega)(z(n) = x(n) \cdot_{Ra} y(n))
). Precedence 20.
```

DEFINITION FS.5.39: Infix function $\cdot_{\mathtt{SeqRa}}$. $x \cdot_{\mathtt{SeqRa}} y \simeq (!z)(x, y, z \in \mathtt{Seq}_{\mathtt{Ra}} \wedge (\forall n \in \omega)(z(n) = x(n) \cdot_{Ra} y(n)))$. Precedence 20.

$\cdot_{\mathtt{SeqRa}}(x, y) \simeq (\iota y_0)(\exists z)(y_0 = z \wedge (x \in \mathtt{Seq}_{\mathtt{Ra}} \wedge y \in \mathtt{Seq}_{\mathtt{Ra}} \wedge z \in \mathtt{Seq}_{\mathtt{Ra}} \wedge (\forall n)(n \in \omega \rightarrow (\iota x_0)(\varpi_{\mathtt{0}}(n, x_0) \in z) = \cdot_{\mathtt{Ra}}((\iota x_0)(\varpi_{\mathtt{0}}(n, x_0) \in x), (\iota x_0)(\varpi_{\mathtt{0}}(n, x_0) \in y)))))$

```
DEFINITION FS.5.40: Infix relation <_{N}. x <_{N} y \iff
x,y \in \omega \wedge x <_{O} y.
```

DEFINITION FS.5.40: Infix relation $<_{\mathtt{N}}$. $x <_{\mathtt{N}} y \leftrightarrow x, y \in \omega \wedge x <_{\mathtt{O}} y$.

$<_{\mathtt{N}}[x, y] \leftrightarrow x \in \omega \wedge y \in \omega \wedge <_{\mathtt{O}}[x, y]$

```
DEFINITION FS.5.41: Infix relation >_{N}. x >_{N} y \iff
x,y \in \omega \wedge x >_{O} y.
```

DEFINITION FS.5.41: Infix relation $>_{\mathtt{N}}$. $x >_{\mathtt{N}} y \leftrightarrow x, y \in \omega \wedge x >_{\mathtt{O}} y$.

$>_{\mathtt{N}}[x, y] \leftrightarrow x \in \omega \wedge y \in \omega \wedge >_{\mathtt{O}}[x, y]$

```
DEFINITION FS.5.42: Infix relation \leq_{N}. x \leq_{N} y \iff
x <_{N} y \vee x = y.
```

DEFINITION FS.5.42: Infix relation $\leq_{\mathtt{N}}$. $x \leq_{\mathtt{N}} y \leftrightarrow x <_{\mathtt{N}} y \vee x = y$.

$\leq_{\mathtt{N}}[x, y] \leftrightarrow <_{\mathtt{N}}[x, y] \vee x = y$

```
DEFINITION FS.5.43: Infix relation \geq_{N}. x \geq_{N} y \iff
x >_{N} y \vee x = y.
```

DEFINITION FS.5.43: Infix relation $\geq_{\mathtt{N}}$. $x \geq_{\mathtt{N}} y \leftrightarrow x >_{\mathtt{N}} y \vee x = y$.

$\geq_{\mathtt{N}}[x, y] \leftrightarrow >_{\mathtt{N}}[x, y] \vee x = y$

```
DEFINITION FS.5.44: 0-ary function Cseq_{Ra}. Cseq_{Ra} \simeq {
  x \in Seq_{Ra} : (\forall \veps >_{Ra} 0_{Ra})(\exists n \in \omega)
  (\forall m,r >_{N} n)(
    Av_{Ra}(x(m) -_{Ra} x(r)) <_{Ra} \veps
  )
}.
```

DEFINITION FS.5.44: 0-ary function $\mathtt{Cseq}_{\mathtt{Ra}}$. $\mathtt{Cseq}_{\mathtt{Ra}} \simeq \{x \in \mathtt{Seq}_{\mathtt{Ra}} : (\forall \varepsilon >_{\mathtt{Ra}} 0_{\mathtt{Ra}})(\exists n \in \omega)(\forall m, r >_{\mathtt{N}} n)(\mathtt{Av}_{\mathtt{Ra}}(x(m) -_{Ra} x(r)) <_{\mathtt{Ra}} \varepsilon)\}$.

$\mathtt{Cseq}_{\mathtt{Ra}} \simeq (\iota z_0)(\forall y_0)(y_0 \in z_0 \leftrightarrow (\exists x)(y_0 = x \wedge ((\forall \varepsilon)(>_{\mathtt{Ra}}[\varepsilon, 0_{\mathtt{Ra}}] \rightarrow (\exists n)(n \in \omega \wedge (\forall m, r)(>_{\mathtt{N}}[m, n] \wedge >_{\mathtt{N}}[r, n] \rightarrow <_{\mathtt{Ra}}[\mathtt{Av}_{\mathtt{Ra}}(-_{\mathtt{Ra}}((\iota x_0)(\varpi_0(m, x_0) \in x), (\iota x_0)(\varpi_0(r, x_0) \in x))), \varepsilon])))) \wedge (x \in \mathtt{Seq}_{\mathtt{Ra}})))$

```
DEFINITION FS.5.45: Infix relation \equiv_{CseqRa}.
x \equiv_{CseqRa} y \iff
(\forall \veps >_{Ra} 0_{Ra})(\exists n \in \omega)
(\forall m >_{N} n)(
    Av_{Ra}(x(m) -_{Ra} y(m)) <_{Ra} \veps
).
```

DEFINITION FS.5.45: Infix relation $\equiv_{\mathtt{CseqRa}}$. $x \equiv_{\mathtt{CseqRa}} y \leftrightarrow (\forall \varepsilon >_{\mathtt{Ra}} 0_{\mathtt{Ra}})(\exists n \in \omega)(\forall m >_{\mathtt{N}} n)(\mathtt{Av}_{\mathtt{Ra}}(x(m) -_{Ra} y(m)) <_{\mathtt{Ra}} \varepsilon)$.

$\equiv_{\mathtt{CseqRa}}[x, y] \leftrightarrow (\forall \varepsilon)(>_{\mathtt{Ra}}[\varepsilon, 0_{\mathtt{Ra}}] \rightarrow (\exists n)(n \in \omega \wedge (\forall m)(>_{\mathtt{N}}[m, n] \rightarrow <_{\mathtt{Ra}}[\mathtt{Av}_{\mathtt{Ra}}(-_{\mathtt{Ra}}((\iota x_0)(\varpi_0(m, x_0) \in x), (\iota x_0)(\varpi_0(m, x_0) \in y))), \varepsilon])))$

```
DEFINITION FS.5.46: Infix relation <_{CseqRa}.
x <_{CseqRa} y \iff x,y \in Cseq_{Ra}
\wedge (\exists \delta >_{Ra} 0_{Ra})(\exists n \in \omega)
(\forall m >_{N} n)(
  x(m) +_{Ra} \delta <_{Ra} y(m)
).
```

DEFINITION FS.5.46: Infix relation $<_{\texttt{CseqRa}}$. $x <_{\texttt{CseqRa}} y \leftrightarrow x, y \in \texttt{Cseq}_{\texttt{Ra}} \wedge (\exists \delta >_{\texttt{Ra}} 0_{\texttt{Ra}})$ $(\exists n \in \omega)(\forall m >_{\texttt{N}} n)(x(m) +_{Ra} \delta <_{\texttt{Ra}} y(m))$.

$<_{\texttt{CseqRa}}[x, y] \leftrightarrow x \in \texttt{Cseq}_{\texttt{Ra}} \wedge y \in \texttt{Cseq}_{\texttt{Ra}} \wedge (\exists \delta)(>_{\texttt{Ra}}[\delta, 0_{\texttt{Ra}}] \wedge (\exists n)(n \in \omega \wedge (\forall m)(>_{\texttt{N}}[m, n] \rightarrow$ $<_{\texttt{Ra}}[+_{\texttt{Ra}}((\iota x_0)(\varpi_0(m, x_0) \in x), \delta), (\iota x_0)(\varpi_0(m, x_0) \in y)])))$

```
DEFINITION FS.5.47: 0-ary function \mathbb{R}. \mathbb{R} \simeq {
  Coset(x,{<u,v> : u \equiv_{CseqRa} v}) : x \in Cseq_{Ra}
}.
```

DEFINITION FS.5.47: 0-ary function $\mathbb{R}$. $\mathbb{R} \simeq \{\texttt{Coset}(x, \{\langle u, v\rangle : u \equiv_{\texttt{CseqRa}} v\}) : x \in \texttt{Cseq}_{\texttt{Ra}}\}$.

$\mathbb{R} \simeq (\iota x_1)(\forall z_0)(z_0 \in x_1 \leftrightarrow (\exists x)(z_0 = \texttt{Coset}(x, (\iota y_0)(\forall x_0)(x_0 \in y_0 \leftrightarrow (\exists u, v)(x_0 =$ $\varpi_0(u, v) \wedge (\equiv_{\texttt{CseqRa}}[u, v])))) \wedge (x \in \texttt{Cseq}_{\texttt{Ra}})))$

```
DEFINITION FS.5.48.1: Infix relation <_{R}. x <_{R} y \iff
(\exists z,w)(
  x,y \in \mathbb{R} \wedge z \in x \wedge
  w \in y \wedge z <_{CseqRa} w
).
```

DEFINITION FS.5.48.1: Infix relation $<_{\texttt{R}}$. $x <_{\texttt{R}} y \leftrightarrow (\exists z, w)(x, y \in \mathbb{R} \wedge z \in x \wedge w \in y \wedge$ $z <_{\texttt{CseqRa}} w)$.

$<_{\texttt{R}}[x, y] \leftrightarrow (\exists z, w)x \in \mathbb{R} \wedge y \in \mathbb{R} \wedge z \in x \wedge w \in y \wedge <_{\texttt{CseqRa}}[z, w]$

```
DEFINITION FS.5.48.2: Infix relation >_{R}. x >_{R} y \iff y <_{R} x.
```

DEFINITION FS.5.48.2: Infix relation $>_{\mathtt{R}}$. $x >_{\mathtt{R}} y \leftrightarrow y <_{\mathtt{R}} x$.

$>_{\mathtt{R}}[x, y] \leftrightarrow <_{\mathtt{R}}[y, x]$

```
DEFINITION FS.5.48.3: Infix relation \leq_{R}. x \leq_{R} y \iff
x <_{R} y \vee x = y.
```

DEFINITION FS.5.48.3: Infix relation $\leq_{\mathtt{R}}$. $x \leq_{\mathtt{R}} y \leftrightarrow x <_{\mathtt{R}} y \vee x = y$.

$\leq_{\mathtt{R}}[x, y] \leftrightarrow <_{\mathtt{R}}[x, y] \vee x = y$

```
DEFINITION FS.5.48.4: Infix relation \geq_{R}. x \geq_{R} y \iff
x >_{R} y \vee x = y.
```

DEFINITION FS.5.48.4: Infix relation $\geq_{\mathtt{R}}$. $x \geq_{\mathtt{R}} y \leftrightarrow x >_{\mathtt{R}} y \vee x = y$.

$\geq_{\mathtt{R}}[x, y] \leftrightarrow >_{\mathtt{R}}[x, y] \vee x = y$

```
DEFINITION FS.5.49: Infix function +_{R}. x +_{R} y \simeq (!z)(
  x,y,z \in \mathbb{R} \wedge (\exists a,b,c)(
    a \in x \wedge b \in y \wedge c \in z \wedge
    a +_{SeqRa} b \equiv_{CseqRa} c
  )
). Precedence 40.
```

DEFINITION FS.5.49: Infix function $+_{\mathtt{R}}$. $x +_{\mathtt{R}} y \simeq (!z)(x, y, z \in \mathbb{R} \wedge (\exists a, b, c)(a \in x \wedge b \in y \wedge c \in z \wedge a +_{SeqRa} b \equiv_{\mathtt{CseqRa}} c))$. Precedence 40.

$+_{\mathtt{R}}(x, y) \simeq (\iota x_0)(\exists z)(x_0 = z \wedge (x \in \mathbb{R} \wedge y \in \mathbb{R} \wedge z \in \mathbb{R} \wedge (\exists a, b, c) a \in x \wedge b \in y \wedge c \in z \wedge \equiv_{\mathtt{CseqRa}}[+_{\mathtt{SeqRa}}(a, b), c]))$

```
DEFINITION FS.5.49.A: Infix function -_{R}. x -_{R} y \simeq
(!z)(x = y +_{R} z). Precedence 60.
```

DEFINITION FS.5.49.A: Infix function $-_{\mathtt{R}}$. $x -_{\mathtt{R}} y \simeq (!z)(x = y +_R z)$. Precedence 60.

$-_{\mathtt{R}}(x, y) \simeq (\iota x_0)(\exists z)(x_0 = z \wedge (x = +_{\mathtt{R}}(y, z)))$

```
DEFINITION FS.5.50: Infix function \cdot_{R}.
x \cdot_{R} y \simeq (!z)(
  x,y,z \in \mathbb{R} \wedge (\exists a,b,c)(
    a \in x \wedge b \in y \wedge c \in z \wedge
    a \cdot_{SeqRa} b \equiv_{CseqRa} c
  )
). Precedence 20.
```

DEFINITION FS.5.50: Infix function $\cdot_{\mathtt{R}}$. $x \cdot_{\mathtt{R}} y \simeq (!z)(x, y, z \in \mathbb{R} \wedge (\exists a, b, c)(a \in x \wedge b \in y \wedge c \in z \wedge a \cdot_{SeqRa} b \equiv_{\mathtt{CseqRa}} c))$. Precedence 20.

$\cdot_{\mathtt{R}}(x, y) \simeq (\iota x_0)(\exists z)(x_0 = z \wedge (x \in \mathbb{R} \wedge y \in \mathbb{R} \wedge z \in \mathbb{R} \wedge (\exists a, b, c) a \in x \wedge b \in y \wedge c \in z \wedge \equiv_{\mathtt{CseqRa}}[\cdot_{\mathtt{SeqRa}}(a, b), c]))$

```
DEFINITION FS.5.51: 0-ary function 0_{R}.
0_{R} \simeq (!x \in \mathbb{R})(
  (\exists w \in x)(\forall n \in \omega)(w(n) = 0_{Ra})
).
```

DEFINITION FS.5.51: 0-ary function $\mathtt{0}_{\mathtt{R}}$. $\mathtt{0}_{\mathtt{R}} \simeq (!x \in \mathbb{R})((\exists w \in x)(\forall n \in \omega)(w(n) = \mathtt{0}_{\mathtt{Ra}}))$.

$\mathtt{0}_{\mathtt{R}} \simeq (\iota y_0)(\exists x)(y_0 = x \wedge ((\exists w)(w \in x \wedge (\forall n)(n \in \omega \rightarrow (\iota x_0)(\varpi_0(n, x_0) \in w) = \mathtt{0}_{\mathtt{Ra}}))) \wedge (x \in \mathbb{R}))$

```
DEFINITION FS.5.52: 0-ary function 1_{R}.
1_{R} \simeq (!x \in \mathbb{R})(
  (\exists w \in x)(\forall n \in \omega)(w(n) = 1_{Ra})
).
```

DEFINITION FS.5.52: 0-ary function $\mathtt{1}_{\mathtt{R}}$. $\mathtt{1}_{\mathtt{R}} \simeq (!x \in \mathbb{R})((\exists w \in x)(\forall n \in \omega)(w(n) = \mathtt{1}_{\mathtt{Ra}}))$.

$\mathtt{1}_{\mathtt{R}} \simeq (\iota y_0)(\exists x)(y_0 = x \wedge ((\exists w)(w \in x \wedge (\forall n)(n \in \omega \rightarrow (\iota x_0)(\varpi_0(n, x_0) \in w) = \mathtt{1}_{\mathtt{Ra}}))) \wedge (x \in \mathbb{R}))$

```
DEFINITION FS.5.53: 1-ary function Av_{R}.
Av_{R}(x) \simeq (!y \in \mathbb{R})(
  (x \geq_{R} 0_{R} \implies y = x) \wedge
  (x <_{R} 0_{R} \implies y = 0_{R} -_{R} x)
).
```

DEFINITION FS.5.53: 1-ary function $\mathtt{Av}_{\mathtt{R}}$. $\mathtt{Av}_{\mathtt{R}}(x) \simeq (!y \in \mathbb{R})((x \geq_{\mathtt{R}} \mathtt{0}_{\mathtt{R}} \rightarrow y = x) \wedge (x <_{\mathtt{R}} \mathtt{0}_{\mathtt{R}} \rightarrow y = \mathtt{0}_{\mathtt{R}} -_R x))$.

$\mathtt{Av}_{\mathtt{R}}(x) \simeq (\iota x_0)(\exists y)(x_0 = y \wedge (\geq_{\mathtt{R}}[x, \mathtt{0}_{\mathtt{R}}] \rightarrow y = x \wedge <_{\mathtt{R}}[x, \mathtt{0}_{\mathtt{R}}] \rightarrow y = -_{\mathtt{R}}(\mathtt{0}_{\mathtt{R}}, x)) \wedge (y \in \mathbb{R}))$

```
DEFINITION FS.5.53.A: 1-ary function Id_{R}.
Id_{R}(x) \simeq (!y \in \mathbb{R})(
  (\exists w \in y)(\forall n \in \omega)(w(n) = x)
).
```

DEFINITION FS.5.53.A: 1-ary function $\mathtt{Id}_{\mathtt{R}}$. $\mathtt{Id}_{\mathtt{R}}(x) \simeq (!y \in \mathbb{R})((\exists w \in y)(\forall n \in \omega)(w(n) = x))$.

$\mathtt{Id}_{\mathtt{R}}(x) \simeq (\iota y_0)(\exists y)(y_0 = y \wedge ((\exists w)(w \in y \wedge (\forall n)(n \in \omega \rightarrow (\iota x_0)(\varpi_0(n, x_0) \in w) = x))) \wedge (y \in \mathbb{R}))$

```
DEFINITION FS.5.53.B: 1-ary function Incl_{NraR}.
If x \in Nra then Incl_{NraR}(x) \simeq Id_{R}(Incl_{NraRa}(x)).
```

DEFINITION FS.5.53.B: 1-ary function $\texttt{Incl}_{\texttt{NraR}}$. If $x \in \texttt{Nra}$ then $\texttt{Incl}_{\texttt{NraR}}(x) \simeq \texttt{Id}_{\texttt{R}}(\texttt{Incl}_{\texttt{NraRa}}(x))$.

$\texttt{Incl}_{\texttt{NraR}}(x) \simeq (\iota x_0)(x \in \texttt{Nra} \wedge x_0 \simeq \texttt{Id}_{\texttt{R}}(\texttt{Incl}_{\texttt{NraRa}}(x)))$

```
DEFINITION FS.5.53.C: 1-ary function Incl_{FrR}.
If x \in Fr then Incl_{FrR}(x) \simeq Id_{R}(Incl_{FrRa}(x)).
```

DEFINITION FS.5.53.C: 1-ary function $\texttt{Incl}_{\texttt{FrR}}$. If $x \in \texttt{Fr}$ then $\texttt{Incl}_{\texttt{FrR}}(x) \simeq \texttt{Id}_{\texttt{R}}(\texttt{Incl}_{\texttt{FrRa}}(x))$.

$\texttt{Incl}_{\texttt{FrR}}(x) \simeq (\iota x_0)(x \in \texttt{Fr} \wedge x_0 \simeq \texttt{Id}_{\texttt{R}}(\texttt{Incl}_{\texttt{FrRa}}(x)))$

```
DEFINITION FS.5.54: 0-ary function Ra_{R}.
Ra_{R} \simeq {Id_{R}(x) : x \in Ra}.
```

DEFINITION FS.5.54: 0-ary function $\texttt{Ra}_{\texttt{R}}$. $\texttt{Ra}_{\texttt{R}} \simeq \{\texttt{Id}_{\texttt{R}}(x) : x \in \texttt{Ra}\}$.

$\texttt{Ra}_{\texttt{R}} \simeq (\iota y_0)(\forall x_0)(x_0 \in y_0 \leftrightarrow (\exists x)(x_0 = \texttt{Id}_{\texttt{R}}(x) \wedge (x \in \texttt{Ra})))$

```
DEFINITION FS.5.55: 0-ary function Seq_{R}.
Seq_{R} \simeq Maps(\omega,\mathbb{R}).
```

DEFINITION FS.5.55: 0-ary function $\texttt{Seq}_{\texttt{R}}$. $\texttt{Seq}_{\texttt{R}} \simeq \texttt{Maps}(\omega, \mathbb{R})$.

$\texttt{Seq}_{\texttt{R}} \simeq \texttt{Maps}(\omega, \mathbb{R})$

```
DEFINITION FS.5.56: 0-ary function Cseq_{R}. Cseq_{R} \simeq {
  x \in Seq_{R} : (\forall \veps >_{Ra} 0_{Ra})(\exists n \in \omega)
  (\forall m,r >_{N} n)(
    Av_{R}(x(m) -_{R} x(r)) <_{R} \veps
  )
}.
```

DEFINITION FS.5.56: 0-ary function $\texttt{Cseq}_{\texttt{R}}$. $\texttt{Cseq}_{\texttt{R}} \simeq \{x \in \texttt{Seq}_{\texttt{R}} : (\forall \varepsilon >_{\texttt{Ra}} \texttt{0}_{\texttt{Ra}})(\exists n \in \omega)(\forall m, r >_{\texttt{N}} n)(\texttt{Av}_{\texttt{R}}(x(m) -_R x(r)) <_{\texttt{R}} \varepsilon)\}$.

$\texttt{Cseq}_{\texttt{R}} \simeq (\iota z_0)(\forall y_0)(y_0 \in z_0 \leftrightarrow (\exists x)(y_0 = x \wedge ((\forall \varepsilon)(>_{\texttt{Ra}}[\varepsilon, \texttt{0}_{\texttt{Ra}}] \rightarrow (\exists n)(n \in \omega \wedge (\forall m, r)(>_{\texttt{N}}[m, n] \wedge >_{\texttt{N}}[r, n] \rightarrow <_{\texttt{R}}[\texttt{Av}_{\texttt{R}}(-_{\texttt{R}}((\iota x_0)(\varpi_0(m, x_0) \in x), (\iota x_0)(\varpi_0(r, x_0) \in x))), \varepsilon])))) \wedge (x \in \texttt{Seq}_{\texttt{R}})))$

```
DEFINITION FS.5.57: 1-ary function Lim.
If x \in Seq_{R} then Lim(x) \simeq
(!y \in \mathbb{R})(
  (\forall \veps >_{R} 0_{R})(\exists n \in \omega)
  (\forall m >_{N} n)(
    Av_{R}(x(m) -_{R} y) <_{R} \veps
  )
).
```

DEFINITION FS.5.57: 1-ary function $\texttt{Lim}$. If $x \in \texttt{Seq}_{\texttt{R}}$ then $\texttt{Lim}(x) \simeq (!y \in \mathbb{R})((\forall \varepsilon >_{\texttt{R}} \texttt{0}_{\texttt{R}})(\exists n \in \omega)(\forall m >_{\texttt{N}} n)(\texttt{Av}_{\texttt{R}}(x(m) -_R y) <_{\texttt{R}} \varepsilon))$.

$\texttt{Lim}(x) \simeq (\iota z_0)(x \in \texttt{Seq}_{\texttt{R}} \wedge z_0 \simeq (\iota y_0)(\exists y)(y_0 = y \wedge ((\forall \varepsilon)(>_{\texttt{R}}[\varepsilon, \texttt{0}_{\texttt{R}}] \rightarrow (\exists n)(n \in \omega \wedge (\forall m)(>_{\texttt{N}}[m, n] \rightarrow <_{\texttt{R}}[\texttt{Av}_{\texttt{R}}(-_{\texttt{R}}((\iota x_0)(\varpi_0(m, x_0) \in x), y)), \varepsilon])))) \wedge (y \in \mathbb{R})))$

```
DEFINITION FS.5.58: 2-ary relation UB_{R}.
UB_{R}[x,A] \iff x \in \mathbb{R} \wedge
A \subseteq \mathbb{R} \wedge (\forall y \in A)(y \leq_{R} x).
```

DEFINITION FS.5.58: 2-ary relation $\texttt{UB}_{\texttt{R}}$. $\texttt{UB}_{\texttt{R}}[x, A] \leftrightarrow x \in \mathbb{R} \wedge A \subseteq \mathbb{R} \wedge (\forall y \in A)(y \leq_{\texttt{R}} x)$.

$\texttt{UB}_{\texttt{R}}[x, A] \leftrightarrow x \in \mathbb{R} \wedge \subseteq[A, \mathbb{R}] \wedge (\forall y)(y \in A \rightarrow \leq_{\texttt{R}}[y, x])$

```
DEFINITION FS.5.59: 1-ary function Min_{R}.
If A \subseteq \mathbb{R} then
Min_{R}(A) \simeq (!x \in A)(
  (\forall y \in A)(\neg(y <_{R} x))
).
```

DEFINITION FS.5.59: 1-ary function $\texttt{Min}_{\texttt{R}}$. If $A \subseteq \mathbb{R}$ then $\texttt{Min}_{\texttt{R}}(A) \simeq (!x \in A)((\forall y \in A)(\neg(y <_{\texttt{R}} x)))$.

$\texttt{Min}_{\texttt{R}}(A) \simeq (\iota y_0)(\subseteq[A, \mathbb{R}] \wedge y_0 \simeq (\iota x_0)(\exists x)(x_0 = x \wedge ((\forall y)(y \in A \rightarrow \neg(<_{\texttt{R}}[y, x]))) \wedge (x \in A)))$

```
DEFINITION FS.5.59.5: 1-ary function Max_{R}.
If A \subseteq \mathbb{R} then
Max_{R}(A) \simeq (!x \in A)(
  (\forall y \in A)(\neg(x <_{R} y))
).
```

DEFINITION FS.5.59.5: 1-ary function $\texttt{Max}_{\texttt{R}}$. If $A \subseteq \mathbb{R}$ then $\texttt{Max}_{\texttt{R}}(A) \simeq (!x \in A)((\forall y \in A)(\neg(x <_{\texttt{R}} y)))$.

$\texttt{Max}_{\texttt{R}}(A) \simeq (\iota y_0)(\subseteq[A, \mathbb{R}] \wedge y_0 \simeq (\iota x_0)(\exists x)(x_0 = x \wedge ((\forall y)(y \in A \rightarrow \neg(<_{\texttt{R}}[x, y]))) \wedge (x \in A)))$

```
DEFINITION FS.5.60: 1-ary function Lub_{R}.
If A \subseteq \mathbb{R} then
Lub_{R}(A) \simeq Min_{R}({x : UB_{R}[x,A]}).
```

DEFINITION FS.5.60: 1-ary function $\texttt{Lub}_{\texttt{R}}$. If $A \subseteq \mathbb{R}$ then $\texttt{Lub}_{\texttt{R}}(A) \simeq \texttt{Min}_{\texttt{R}}(\{x : \texttt{UB}_{\texttt{R}}[x, A]\})$.

$\texttt{Lub}_{\texttt{R}}(A) \simeq (\iota z_0)(\subseteq[A, \mathbb{R}] \wedge z_0 \simeq \texttt{Min}_{\texttt{R}}((\iota y_0)(\forall x_0)(x_0 \in y_0 \leftrightarrow (\exists x)(x_0 = x \wedge (\texttt{UB}_{\texttt{R}}[x, A])))))$

```
DEFINITION FS.5.61.pre: 1-ary function Finitesumset_{R}.
If (\exists n \in \omega)
(FCN[f,n,\mathbb{R}]) then Finitesumset_{R}(f) \simeq (!x)(
  (\forall m \in \omega)(
    (m <_{N} n \implies (
       x(0_{N}) = 0_{R} \wedge
       x(Suc(m)) = x(m) +_{R} f(Suc(m))
```

```
     )
    ) \wedge
    (m \geq_{N} n \implies x(Suc(m)) = 0_{R})
  ) \wedge
  (\forall m)(
    x(m) \dn \iff m \in \omega
  )
).
```

DEFINITION FS.5.61.pre: 1-ary function $\texttt{Finitesumset}_{\texttt{R}}$. If $(\exists n \in \omega)(\texttt{FCN}[f, n, \mathbb{R}])$ then $\texttt{Finitesumset}_{\texttt{R}}(f) \simeq (!x)((\forall m \in \omega)((m <_{\texttt{N}} n \rightarrow (x(\texttt{0}_{\texttt{N}}) = \texttt{0}_{\texttt{R}} \wedge x(\texttt{Suc}(m)) = x(m) +_R f(\texttt{Suc}(m)))) \wedge (m \geq_{\texttt{N}} n \rightarrow x(\texttt{Suc}(m)) = \texttt{0}_{\texttt{R}})) \wedge (\forall m)(x(m)\downarrow \leftrightarrow m \in \omega))$.

$\texttt{Finitesumset}_{\texttt{R}}(f) \simeq (\iota z_0)((\exists n)(n \in \omega \wedge \texttt{FCN}[f, n, \mathbb{R}]) \wedge z_0 \simeq (\iota y_0)(\exists x)(y_0 = x \wedge ((\forall m)(m \in \omega \rightarrow <_{\texttt{N}}[m, n] \rightarrow (\iota x_0)(\varpi_{\texttt{0}}(\texttt{0}_{\texttt{N}}, x_0) \in x) = \texttt{0}_{\texttt{R}} \wedge (\iota x_0)(\varpi_{\texttt{0}}(\texttt{Suc}(m), x_0) \in x) = +_{\texttt{R}}((\iota x_0)(\varpi_{\texttt{0}}(m, x_0) \in x), (\iota x_0)(\varpi_{\texttt{0}}(\texttt{Suc}(m), x_0) \in f)) \wedge \geq_{\texttt{N}}[m, n] \rightarrow (\iota x_0)(\varpi_{\texttt{0}}(\texttt{Suc}(m), x_0) \in x) = \texttt{0}_{\texttt{R}}) \wedge (\forall m)(\iota x_0)(\varpi_{\texttt{0}}(m, x_0) \in x)\downarrow \leftrightarrow m \in \omega)))$

```
DEFINITION FS.5.61: 1-ary function Finitesum_{R}.
If (\exists n \in \omega)
(FCN[f,n,\mathbb{R}]) then Finitesum_{R}(f) \simeq
(!r \in \mathbb{R})(<Dom(f),r> \in Finitesumset_{R}(f)).
```

DEFINITION FS.5.61: 1-ary function $\texttt{Finitesum}_{\texttt{R}}$. If $(\exists n \in \omega)(\texttt{FCN}[f, n, \mathbb{R}])$ then $\texttt{Finitesum}_{\texttt{R}}(f) \simeq (!r \in \mathbb{R})(\langle \texttt{Dom}(f), r\rangle \in \texttt{Finitesumset}_{\texttt{R}}(f))$.

$\texttt{Finitesum}_{\texttt{R}}(f) \simeq (\iota y_0)((\exists n)(n \in \omega \wedge \texttt{FCN}[f, n, \mathbb{R}]) \wedge y_0 \simeq (\iota x_0)(\exists r)(x_0 = r \wedge (\varpi_{\texttt{0}}(\texttt{Dom}(f), r) \in \texttt{Finitesumset}_{\texttt{R}}(f)) \wedge (r \in \mathbb{R})))$

```
DEFINITION FS.5.62: 1-ary function Sqrt_{R}.
If r \in \mathbb{R} then Sqrt_{R}(r)
\simeq (!y \in \mathbb{R})(
  y \geq_{R} 0_{R} \wedge y \cdot_{R} y = r
).
```

DEFINITION FS.5.62: 1-ary function $\texttt{Sqrt}_{\texttt{R}}$. If $r \in \mathbb{R}$ then $\texttt{Sqrt}_{\texttt{R}}(r) \simeq (!y \in \mathbb{R})(y \geq_{\texttt{R}} 0_{\texttt{R}} \wedge y \cdot_R y = r)$.

$\texttt{Sqrt}_{\texttt{R}}(r) \simeq (\iota y_0)(r \in \mathbb{R} \wedge y_0 \simeq (\iota x_0)(\exists y)(x_0 = y \wedge (\geq_{\texttt{R}}[y, 0_{\texttt{R}}] \wedge \cdot_{\texttt{R}}(y,y) = r) \wedge (y \in \mathbb{R})))$

```
DEFINITION FS.5.63: 1-ary function Sup_{R}. Sup_{R}(A) \simeq (!s)(
SUP[s,{<x,y>: x <_{R} y},A]
).
```

DEFINITION FS.5.63: 1-ary function $\texttt{Sup}_{\texttt{R}}$. $\texttt{Sup}_{\texttt{R}}(A) \simeq (!s)(\texttt{SUP}[s, \{\langle x, y\rangle : x <_{\texttt{R}} y\}, A])$.

$\texttt{Sup}_{\texttt{R}}(A) \simeq (\iota z_0)(\exists s)(z_0 = s \wedge (\texttt{SUP}[s, (\iota y_0)(\forall x_0)(x_0 \in y_0 \leftrightarrow (\exists x, y)(x_0 = \varpi_0(x,y) \wedge (<_{\texttt{R}}[x,y]))), A]))$

```
DEFINITION FS.5.64: 1-ary function Inf_{R}. Inf_{R}(A) \simeq (!g)(
INF[g,{<x,y> : x <_{R} y},A]
).
```

DEFINITION FS.5.64: 1-ary function $\texttt{Inf}_{\texttt{R}}$. $\texttt{Inf}_{\texttt{R}}(A) \simeq (!g)(\texttt{INF}[g, \{\langle x, y\rangle : x <_{\texttt{R}} y\}, A])$.

$\texttt{Inf}_{\texttt{R}}(A) \simeq (\iota z_0)(\exists g)(z_0 = g \wedge (\texttt{INF}[g, (\iota y_0)(\forall x_0)(x_0 \in y_0 \leftrightarrow (\exists x, y)(x_0 = \varpi_0(x,y) \wedge (<_{\texttt{R}}[x,y]))), A]))$

Appendix G

Definitions: Topology – Munkres

For each definition (with a few exceptions) from Munkres's *Topology* [12], Sections 12 through 38, find below a triplet, consisting of:

i. A transcription of the definition into LPT input,

ii. The LaTeX version of the input, and

iii. The DZFC translation of the definition, as performed by `lpt2dzfc`.

Side-by-side comparison of these definitions against the originals as they appear in [12] will reveal how well the language LPT allows an approximation to the original.

G.1 Section 12

```
DEFINITION MunkTop.12.1: 2-ary relation TOPOLOGY.
TOPOLOGY[\mathscr{T},X]
\iff \mathscr{T} \subseteq \wp(X)
\wedge \varnothing \in \mathscr{T} \wedge X \in \mathscr{T} \wedge
(\forall S \subseteq \mathscr{T})(\cup(S) \in \mathscr{T}) \wedge
(\forall U,V \in \mathscr{T})(U \cap V \in \mathscr{T}).
```

DEFINITION MunkTop.12.1: 2-ary relation `TOPOLOGY`. $\texttt{TOPOLOGY}[\mathscr{T}, X] \leftrightarrow \mathscr{T} \subseteq \wp(X) \wedge \varnothing \in \mathscr{T} \wedge X \in \mathscr{T} \wedge (\forall S \subseteq \mathscr{T})(\cup(S) \in \mathscr{T}) \wedge (\forall U, V \in \mathscr{T})(U \cap V \in \mathscr{T})$.

$\texttt{TOPOLOGY}[\mathscr{T}, X] \leftrightarrow \subseteq[\mathscr{T}, \wp(X)] \wedge \varnothing \in \mathscr{T} \wedge X \in \mathscr{T} \wedge (\forall S)(\subseteq[S, \mathscr{T}] \rightarrow \cup(S) \in \mathscr{T}) \wedge (\forall U, V)(U \in \mathscr{T} \wedge V \in \mathscr{T} \rightarrow \cap(U, V) \in \mathscr{T})$

```
DEFINITION MunkTop.12.2: 2-ary relation TOPSP. TOPSP[X,\mathscr{T}] \iff
TOPOLOGY[\mathscr{T},X].
```

DEFINITION MunkTop.12.2: 2-ary relation TOPSP. $\texttt{TOPSP}[X, \mathscr{T}] \leftrightarrow \texttt{TOPOLOGY}[\mathscr{T}, X]$.

$\texttt{TOPSP}[X, \mathscr{T}] \leftrightarrow \texttt{TOPOLOGY}[\mathscr{T}, X]$

```
DEFINITION MunkTop.12.3: 3-ary relation OPENSET. OPENSET[U,X,\mathscr{T}]
\iff TOPSP[X,\mathscr{T}] \wedge U \in \mathscr{T}.
```

DEFINITION MunkTop.12.3: 3-ary relation OPENSET. $\texttt{OPENSET}[U, X, \mathscr{T}] \leftrightarrow \texttt{TOPSP}[X, \mathscr{T}] \wedge U \in \mathscr{T}$.

$\texttt{OPENSET}[U, X, \mathscr{T}] \leftrightarrow \texttt{TOPSP}[X, \mathscr{T}] \wedge U \in \mathscr{T}$

```
DEFINITION MunkTop.12.4.a: 3-ary relation FINERTOP. If TOPSP[X,\mathscr{T}]
\wedge TOPSP[X,\mathscr{T}'] then FINERTOP[\mathscr{T}',\mathscr{T},X] \iff
\mathscr{T}' \supseteq \mathscr{T}.
```

DEFINITION MunkTop.12.4.a: 3-ary relation FINERTOP. If $\texttt{TOPSP}[X, \mathscr{T}] \wedge \texttt{TOPSP}[X, \mathscr{T}']$ then $\texttt{FINERTOP}[\mathscr{T}', \mathscr{T}, X] \leftrightarrow \mathscr{T}' \supseteq \mathscr{T}$.

$\texttt{FINERTOP}[\mathscr{T}', \mathscr{T}, X] \leftrightarrow (\texttt{TOPSP}[X, \mathscr{T}] \wedge \texttt{TOPSP}[X, \mathscr{T}'] \wedge \supseteq[\mathscr{T}', \mathscr{T}])$

```
DEFINITION MunkTop.12.4.b: 3-ary relation STRFINERTOP.
If TOPSP[X,\mathscr{T}] \wedge TOPSP[X,\mathscr{T}'] then
STRFINERTOP[\mathscr{T}',\mathscr{T},X] \iff
\mathscr{T}' \supseteq \mathscr{T} \wedge
\mathscr{T}' \neq \mathscr{T}.
```

DEFINITION MunkTop.12.4.b: 3-ary relation STRFINERTOP. If TOPSP$[X, \mathscr{T}]\wedge$ TOPSP$[X, \mathscr{T}']$ then STRFINERTOP$[\mathscr{T}', \mathscr{T}, X] \leftrightarrow \mathscr{T}' \supseteq \mathscr{T} \wedge \mathscr{T}' \neq \mathscr{T}$.

STRFINERTOP$[\mathscr{T}', \mathscr{T}, X] \leftrightarrow$ (TOPSP$[X, \mathscr{T}] \wedge$ TOPSP$[X, \mathscr{T}'] \wedge \supseteq[\mathscr{T}', \mathscr{T}] \wedge \neg(\mathscr{T}' = \mathscr{T})$)

```
DEFINITION MunkTop.12.4.c: 3-ary relation COARSERTOP.
If TOPSP[X,\mathscr{T}] \wedge TOPSP[X,\mathscr{T}'] then
COARSERTOP[\mathscr{T}',\mathscr{T},X] \iff
\mathscr{T}' \subseteq \mathscr{T}.
```

DEFINITION MunkTop.12.4.c: 3-ary relation COARSERTOP. If TOPSP$[X, \mathscr{T}]\wedge$TOPSP$[X, \mathscr{T}']$ then COARSERTOP$[\mathscr{T}', \mathscr{T}, X] \leftrightarrow \mathscr{T}' \subseteq \mathscr{T}$.

COARSERTOP$[\mathscr{T}', \mathscr{T}, X] \leftrightarrow$ (TOPSP$[X, \mathscr{T}] \wedge$ TOPSP$[X, \mathscr{T}'] \wedge \subseteq[\mathscr{T}', \mathscr{T}]$)

```
DEFINITION MunkTop.12.4.d: 3-ary relation STRCOARSERTOP.
If TOPSP[X,\mathscr{T}] \wedge TOPSP[X,\mathscr{T}'] then
STRCOARSERTOP[\mathscr{T}',\mathscr{T},X] \iff
\mathscr{T}' \subseteq \mathscr{T} \wedge \mathscr{T}' \neq \mathscr{T}.
```

DEFINITION MunkTop.12.4.d: 3-ary relation STRCOARSERTOP. If TOPSP$[X, \mathscr{T}] \wedge$ TOPSP$[X, \mathscr{T}']$ then STRCOARSERTOP$[\mathscr{T}', \mathscr{T}, X] \leftrightarrow \mathscr{T}' \subseteq \mathscr{T} \wedge \mathscr{T}' \neq \mathscr{T}$.

STRCOARSERTOP$[\mathscr{T}', \mathscr{T}, X] \leftrightarrow$ (TOPSP$[X, \mathscr{T}]\wedge$TOPSP$[X, \mathscr{T}']\wedge\subseteq[\mathscr{T}', \mathscr{T}]\wedge\neg(\mathscr{T}' = \mathscr{T})$)

G.2 Section 13

```
DEFINITION MunkTop.13.1: 2-ary relation TOPBASIS. TOPBASIS[\mathscr{B},X]
\iff \mathscr{B} \subseteq \wp(X) \wedge
(\forall x \in X)(\exists B \in \mathscr{B})(x \in B) \wedge
(\forall x \in X)(\forall B_{1}, B_{2} \in \mathscr{B})
(\exists B_{3} \in \mathscr{B})(
  x \in B_{3} \wedge B_{3} \subseteq B_{1} \cap B_{2}
).
```

DEFINITION MunkTop.13.1: 2-ary relation TOPBASIS. TOPBASIS$[\mathscr{B}, X] \leftrightarrow \mathscr{B} \subseteq \wp(X) \wedge (\forall x \in X)(\exists B \in \mathscr{B})(x \in B) \wedge (\forall x \in X)(\forall B_1, B_2 \in \mathscr{B})(\exists B_3 \in \mathscr{B})(x \in B_3 \wedge B_3 \subseteq B_1 \cap B_2)$.

TOPBASIS$[\mathscr{B}, X] \leftrightarrow \subseteq[\mathscr{B}, \wp(X)] \wedge (\forall x)(x \in X \rightarrow (\exists B)(B \in \mathscr{B} \wedge x \in B)) \wedge (\forall x)(x \in X \rightarrow (\forall B_1, B_2)(B_1 \in \mathscr{B} \wedge B_2 \in \mathscr{B} \rightarrow (\exists B_3)(B_3 \in \mathscr{B} \wedge x \in B_3 \wedge \subseteq[B_3, \cap(B_1, B_2)])))$

```
DEFINITION MunkTop.13.2: 2-ary function Basisgentop.
If TOPBASIS[\mathscr{B},X] then
Basisgentop(\mathscr{B},X) \simeq (!\mathscr{T} \subseteq \wp(X))(
  (\forall U \subseteq X)(U \in \mathscr{T} \iff
    (\forall x \in U)(\exists B \in \mathscr{B})(
      x \in B \wedge B \subseteq U
    )
  )
).
```

DEFINITION MunkTop.13.2: 2-ary function Basisgentop. If TOPBASIS$[\mathscr{B}, X]$ then Basisgentop$(\mathscr{B}, X) \simeq (!\mathscr{T} \subseteq \wp(X))((\forall U \subseteq X)(U \in \mathscr{T} \leftrightarrow (\forall x \in U)(\exists B \in \mathscr{B})(x \in B \wedge B \subseteq U)))$.

Basisgentop$(\mathscr{B}, X) \simeq (\iota y_0)($TOPBASIS$[\mathscr{B}, X] \wedge y_0 \simeq (\iota x_0)(\exists \mathscr{T})(x_0 = \mathscr{T} \wedge ((\forall U)(\subseteq[U, X] \rightarrow U \in \mathscr{T} \leftrightarrow (\forall x)(x \in U \rightarrow (\exists B)(B \in \mathscr{B} \wedge x \in B \wedge \subseteq[B, U])))) \wedge (\subseteq[\mathscr{T}, \wp(X)])))$

```
DEFINITION MunkTop.13.3.a.basis: 0-ary function Stdrealtopbasis.
Stdrealtopbasis \simeq {U \subseteq \mathbb{R} :
  (\exists a,b \in \mathbb{R})(
    U = {x \in \mathbb{R} : a <_{R} x <_{R} b}
  )
}.
```

DEFINITION MunkTop.13.3.a.basis: 0-ary function Stdrealtopbasis.
Stdrealtopbasis $\simeq \{U \subseteq \mathbb{R} : (\exists a, b \in \mathbb{R})(U = \{x \in \mathbb{R} : a <_{\mathtt{R}} x <_{\mathtt{R}} b\})\}$.

Stdrealtopbasis $\simeq (\iota x_1)(\forall z_0)(z_0 \in x_1 \leftrightarrow (\exists U)(z_0 = U \wedge ((\exists a, b)(a \in \mathbb{R} \wedge b \in \mathbb{R} \wedge U = (\iota y_0)(\forall x_0)(x_0 \in y_0 \leftrightarrow (\exists x)(x_0 = x \wedge (<_{\mathtt{R}}[a, x] \wedge <_{\mathtt{R}}[x, b]) \wedge (x \in \mathbb{R})))))) \wedge (\subseteq[U, \mathbb{R}])))$

```
DEFINITION MunkTop.13.3.a: 0-ary function Stdrealtop. Stdrealtop \simeq
Basisgentop(Stdrealtopbasis,\mathbb{R}).
```

DEFINITION MunkTop.13.3.a: 0-ary function `Stdrealtop`. `Stdrealtop` $\simeq$ `Basisgentop`(`Stdrealtopbasis`, $\mathbb{R}$).

`Stdrealtop` $\simeq$ `Basisgentop`(`Stdrealtopbasis`, $\mathbb{R}$)

```
DEFINITION MunkTop.13.3.b: 0-ary function Lowerlimitrealtop.
Lowerlimitrealtop \simeq Basisgentop(
  {U \subseteq \mathbb{R} :
    (\exists a,b \in \mathbb{R})(
      U = {x \in \mathbb{R} : a \leq_{R} x <_{R} b}
    )
  },
  \mathbb{R}
).
```

DEFINITION MunkTop.13.3.b: 0-ary function `Lowerlimitrealtop`.
`Lowerlimitrealtop` $\simeq$ `Basisgentop`($\{U \subseteq \mathbb{R} : (\exists a, b \in \mathbb{R})(U = \{x \in \mathbb{R} : a \leq_{\mathtt{R}} x <_{\mathtt{R}} b\})\}, \mathbb{R}$).

`Lowerlimitrealtop` $\simeq$ `Basisgentop`($(\iota x_1)(\forall z_0)(z_0 \in x_1 \leftrightarrow (\exists U)(z_0 = U \wedge ((\exists a, b)(a \in \mathbb{R} \wedge b \in \mathbb{R} \wedge U = (\iota y_0)(\forall x_0)(x_0 \in y_0 \leftrightarrow (\exists x)(x_0 = x \wedge (\leq_{\mathtt{R}}[a, x] \wedge <_{\mathtt{R}}[x, b]) \wedge (x \in \mathbb{R}))))) \wedge (\subseteq[U, \mathbb{R}]))), \mathbb{R}$)

```
DEFINITION MunkTop.13.3.c: 0-ary function Krealtop. Krealtop \simeq
Basisgentop(
  Stdrealtopbasis \cup
  {V \subseteq \mathbb{R} :
    (\exists W \in Stdrealtopbasis)(
      V = W \less {Incl_{FrR}(1_{N}/n) : n \in \mathbb{N}}
    )
  },
  \mathbb{R}
).
```

DEFINITION MunkTop.13.3.c: 0-ary function Krealtop. Krealtop $\simeq$ Basisgentop(Stdrealtopbasis $\cup\{V \subseteq \mathbb{R} : (\exists W \in$ Stdrealtopbasis$)$ $(V = W \backslash \{\mathtt{Incl}_{\mathtt{FrR}}(1_{\mathtt{N}}/n) : n \in \mathbb{N}\})\}, \mathbb{R})$.

Krealtop $\simeq$ Basisgentop$(\cup($Stdrealtopbasis$, (\iota x_1)(\forall z_0)(z_0 \in x_1 \leftrightarrow (\exists V)(z_0 = V \wedge ((\exists W)(W \in$ Stdrealtopbasis$\wedge V = \backslash(W, (\iota y_0)(\forall x_0)(x_0 \in y_0 \leftrightarrow (\exists n)(x_0 = \mathtt{Incl}_{\mathtt{FrR}}(/(1_{\mathtt{N}}, n)) \wedge (n \in \mathbb{N})))))) \wedge (\subseteq[V, \mathbb{R}])))), \mathbb{R})$

```
DEFINITION MunkTop.13.4.a: 2-ary relation TOPSUBBASIS.
TOPSUBBASIS[\mathscr{S},X]
\iff \mathscr{S} \subseteq \wp(X) \wedge \cup(\mathscr{S}) = X.
```

DEFINITION MunkTop.13.4.a: 2-ary relation TOPSUBBASIS. TOPSUBBASIS$[\mathscr{S}, X] \leftrightarrow \mathscr{S} \subseteq \wp(X) \wedge \cup(\mathscr{S}) = X$.

TOPSUBBASIS$[\mathscr{S}, X] \leftrightarrow \subseteq[\mathscr{S}, \wp(X)] \wedge \cup(\mathscr{S}) = X$

```
DEFINITION MunkTop.13.4.b: 2-ary function Subbasisgentop.
Subbasisgentop(\mathscr{S},X) \simeq {U \subseteq X :
  (\exists \mathscr{A} \subseteq \wp(\mathscr{S}))(
    (\forall A \in \mathscr{A})(DFIN[A]) \wedge
    U = \cup({\cap(S) : S \in A})
  )
}.
```

DEFINITION MunkTop.13.4.b: 2-ary function Subbasisgentop. Subbasisgentop$(\mathscr{S}, X) \simeq \{U \subseteq X : (\exists \mathscr{A} \subseteq \wp(\mathscr{S}))((\forall A \in \mathscr{A})(\mathtt{DFIN}[A]) \wedge U = \cup(\{\cap(S) : S \in A\}))\}$.

Subbasisgentop$(\mathscr{S}, X) \simeq (\iota x_1)(\forall z_0)(z_0 \in x_1 \leftrightarrow (\exists U)(z_0 = U \wedge ((\exists \mathscr{A})(\subseteq[\mathscr{A}, \wp(\mathscr{S})] \wedge (\forall A)(A \in \mathscr{A} \rightarrow \mathtt{DFIN}[A]) \wedge U = \cup((\iota y_0)(\forall x_0)(x_0 \in y_0 \leftrightarrow (\exists S)(x_0 = \cap(S) \wedge (S \in A)))))) \wedge (\subseteq[U, X])))$

G.3 Section 14

```
DEFINITION MunkTop.14.1: 4-ary function Oointerval. If SSORD[R,X]
\wedge a,b \in X then Oointerval(a,b,X,R) \simeq
{x \in X : a \infixrl{R} x \infixrl{R} b}.
```

DEFINITION MunkTop.14.1: 4-ary function `Oointerval`. If `SSORD`$[R, X] \wedge a, b \in X$ then `Oointerval`$(a, b, X, R) \simeq \{x \in X : aRxRb\}$.

`Oointerval`$(a, b, X, R) \simeq (\iota z_0)($`SSORD`$[R, X] \wedge a \in X \wedge b \in X \wedge z_0 \simeq (\iota y_0)(\forall x_0)(x_0 \in y_0 \leftrightarrow (\exists x)(x_0 = x \wedge (\varpi_0(a, x) \in R \wedge \varpi_0(x, b) \in R) \wedge (x \in X))))$

```
DEFINITION MunkTop.14.1.A: 3-ary relation OOINTERVAL. If SSORD[R,X]
\wedge U \subseteq X then OOINTERVAL[U,X,R] \iff
(\exists a,b \in X)(U = Oointerval(a,b,X,R)).
```

DEFINITION MunkTop.14.1.A: 3-ary relation `OOINTERVAL`. If `SSORD`$[R, X] \wedge U \subseteq X$ then `OOINTERVAL`$[U, X, R] \leftrightarrow (\exists a, b \in X)(U =$ `Oointerval`$(a, b, X, R))$.

`OOINTERVAL`$[U, X, R] \leftrightarrow ($`SSORD`$[R, X] \wedge \subseteq[U, X] \wedge (\exists a, b)(a \in X \wedge b \in X \wedge U =$ `Oointerval`$(a, b, X, R)))$

```
DEFINITION MunkTop.14.2: 4-ary function Ocinterval. If SSORD[R,X]
\wedge a,b \in X then Ocinterval(a,b,X,R) \simeq
{x \in X : a \infixrl{R} x \infixrl{R} b \vee x = b}.
```

DEFINITION MunkTop.14.2: 4-ary function `Ocinterval`. If `SSORD`$[R, X] \wedge a, b \in X$ then `Ocinterval`$(a, b, X, R) \simeq \{x \in X : aRxRb \vee x = b\}$.

`Ocinterval`$(a, b, X, R) \simeq (\iota z_0)($`SSORD`$[R, X] \wedge a \in X \wedge b \in X \wedge z_0 \simeq (\iota y_0)(\forall x_0)(x_0 \in y_0 \leftrightarrow (\exists x)(x_0 = x \wedge (\varpi_0(a, x) \in R \wedge \varpi_0(x, b) \in R \vee x = b) \wedge (x \in X))))$

```
DEFINITION MunkTop.14.2.A: 3-ary relation OCINTERVAL. If SSORD[R,X]
\wedge W \subseteq X then OCINTERVAL[W,X,R] \iff
(\exists a,b \in X)(W = Ocinterval(a,b,X,R)).
```

DEFINITION MunkTop.14.2.A: 3-ary relation OCINTERVAL. If SSORD$[R, X] \wedge W \subseteq X$ then OCINTERVAL$[W, X, R] \leftrightarrow (\exists a, b \in X)(W =$ Ocinterval$(a, b, X, R))$.

OCINTERVAL$[W, X, R] \leftrightarrow$ (SSORD$[R, X] \wedge \subseteq [W, X] \wedge (\exists a, b)(a \in X \wedge b \in X \wedge W =$ Ocinterval$(a, b, X, R)))$

```
DEFINITION MunkTop.14.3: 4-ary function Cointerval. If SSORD[R,X]
\wedge a,b \in X then Cointerval(a,b,X,R) \simeq
{x \in X : a \infixrl{R} x \infixrl{R} b \vee x = a}.
```

DEFINITION MunkTop.14.3: 4-ary function Cointerval. If SSORD$[R, X] \wedge a, b \in X$ then Cointerval$(a, b, X, R) \simeq \{x \in X : aRxRb \vee x = a\}$.

Cointerval$(a, b, X, R) \simeq (\iota z_0)($SSORD$[R, X] \wedge a \in X \wedge b \in X \wedge z_0 \simeq (\iota y_0)(\forall x_0)(x_0 \in y_0 \leftrightarrow (\exists x)(x_0 = x \wedge (\varpi_0(a, x) \in R \wedge \varpi_0(x, b) \in R \vee x = a) \wedge (x \in X))))$

```
DEFINITION MunkTop.14.3.A: 3-ary relation COINTERVAL. If SSORD[R,X]
\wedge W \subseteq X then COINTERVAL[W,X,R] \iff
(\exists a,b \in X)(W = Cointerval(a,b,X,R)).
```

DEFINITION MunkTop.14.3.A: 3-ary relation COINTERVAL. If SSORD$[R, X] \wedge W \subseteq X$ then COINTERVAL$[W, X, R] \leftrightarrow (\exists a, b \in X)(W =$ Cointerval$(a, b, X, R))$.

COINTERVAL$[W, X, R] \leftrightarrow$ (SSORD$[R, X] \wedge \subseteq [W, X] \wedge (\exists a, b)(a \in X \wedge b \in X \wedge W =$ Cointerval$(a, b, X, R)))$

```
DEFINITION MunkTop.14.4: 4-ary function Ccinterval. If SSORD[R,X]
\wedge a,b \in X then Ccinterval(a,b,X,R) \simeq
{x \in X : a \infixrl{R} x \infixrl{R} b \vee x = a \vee x = b}.
```

DEFINITION MunkTop.14.4: 4-ary function Ccinterval. If SSORD$[R, X] \wedge a, b \in X$ then Ccinterval$(a, b, X, R) \simeq \{x \in X : aRxRb \vee x = a \vee x = b\}$.

Ccinterval$(a, b, X, R) \simeq (\iota z_0)($SSORD$[R, X] \wedge a \in X \wedge b \in X \wedge z_0 \simeq (\iota y_0)(\forall x_0)(x_0 \in y_0 \leftrightarrow (\exists x)(x_0 = x \wedge (\varpi_0(a, x) \in R \wedge \varpi_0(x, b) \in R \vee x = a \vee x = b) \wedge (x \in X))))$

```
DEFINITION MunkTop.14.4.A: 3-ary relation CCINTERVAL. If SSORD[R,X]
\wedge F \subseteq X then CCINTERVAL[F,X,R] \iff
(\exists a,b \in X)(F = Ccinterval(a,b,X,R)).
```

DEFINITION MunkTop.14.4.A: 3-ary relation `CCINTERVAL`. If $\texttt{SSORD}[R, X] \wedge F \subseteq X$ then $\texttt{CCINTERVAL}[F, X, R] \leftrightarrow (\exists a, b \in X)(F = \texttt{Ccinterval}(a, b, X, R))$.

$\texttt{CCINTERVAL}[F, X, R] \leftrightarrow (\texttt{SSORD}[R, X] \wedge \subseteq [F, X] \wedge (\exists a, b)(a \in X \wedge b \in X \wedge F = \texttt{Ccinterval}(a, b, X, R)))$

```
DEFINITION MunkTop.14.5: 2-ary function Ordertopbasis. If SSORD[R,X] then
Ordertopbasis(X,R) \simeq {U :
  (\exists a,b \in X)(
    U = Oointerval(a,b,X,R) \vee
    FELT[a,R,X] \wedge U = Cointerval(a,b,X,R) \vee
    LELT[b,R,X] \wedge U = Ocinterval(a,b,X,R)
  )
}.
```

DEFINITION MunkTop.14.5: 2-ary function `Ordertopbasis`. If $\texttt{SSORD}[R, X]$ then $\texttt{Ordertopbasis}(X, R) \simeq \{U : (\exists a, b \in X)(U = \texttt{Oointerval}(a, b, X, R) \vee \texttt{FELT}[a, R, X] \wedge U = \texttt{Cointerval}(a, b, X, R) \vee \texttt{LELT}[b, R, X] \wedge U = \texttt{Ocinterval}(a, b, X, R))\}$.

$\texttt{Ordertopbasis}(X, R) \simeq (\iota z_0)(\texttt{SSORD}[R, X] \wedge z_0 \simeq (\iota y_0)(\forall x_0)(x_0 \in y_0 \leftrightarrow (\exists U)(x_0 = U \wedge ((\exists a, b)(a \in X \wedge b \in X \wedge U = \texttt{Oointerval}(a, b, X, R) \vee \texttt{FELT}[a, R, X] \wedge U = \texttt{Cointerval}(a, b, X, R) \vee \texttt{LELT}[b, R, X] \wedge U = \texttt{Ocinterval}(a, b, X, R))))))$

```
DEFINITION MunkTop.14.6: 2-ary function Ordertop. If SSORD[R,X] then
Ordertop(X,R) \simeq Basisgentop(Ordertopbasis(X,R),X).
```

DEFINITION MunkTop.14.6: 2-ary function `Ordertop`. If $\texttt{SSORD}[R, X]$ then $\texttt{Ordertop}(X, R) \simeq \texttt{Basisgentop}(\texttt{Ordertopbasis}(X, R), X)$.

$\texttt{Ordertop}(X, R) \simeq (\iota x_0)(\texttt{SSORD}[R, X] \wedge x_0 \simeq \texttt{Basisgentop}(\texttt{Ordertopbasis}(X, R), X))$

`DEFINITION MunkTop.14.7: 3-ary function Oplusray. If SSORD[R,X] \wedge a \in X then Oplusray(a,X,R) \simeq {x \in X : a \infixrl{R} x}.`

DEFINITION MunkTop.14.7: 3-ary function `Oplusray`. If $\texttt{SSORD}[R, X] \wedge a \in X$ then $\texttt{Oplusray}(a, X, R) \simeq \{x \in X : aRx\}$.

$\texttt{Oplusray}(a, X, R) \simeq (\iota z_0)(\texttt{SSORD}[R, X] \wedge a \in X \wedge z_0 \simeq (\iota y_0)(\forall x_0)(x_0 \in y_0 \leftrightarrow (\exists x)(x_0 = x \wedge (\varpi_0(a, x) \in R) \wedge (x \in X))))$

`DEFINITION MunkTop.14.8: 3-ary function Ominusray. If SSORD[R,X] \wedge a \in X then Ominusray(a,X,R) \simeq {x \in X : x \infixrl{R} a}.`

DEFINITION MunkTop.14.8: 3-ary function `Ominusray`. If $\texttt{SSORD}[R, X] \wedge a \in X$ then $\texttt{Ominusray}(a, X, R) \simeq \{x \in X : xRa\}$.

$\texttt{Ominusray}(a, X, R) \simeq (\iota z_0)(\texttt{SSORD}[R, X] \wedge a \in X \wedge z_0 \simeq (\iota y_0)(\forall x_0)(x_0 \in y_0 \leftrightarrow (\exists x)(x_0 = x \wedge (\varpi_0(x, a) \in R) \wedge (x \in X))))$

`DEFINITION MunkTop.14.9: 3-ary function Cplusray. If SSORD[R,X] \wedge a \in X then Cplusray(a,X,R) \simeq {x \in X : a \infixrl{R} x \vee x = a}.`

DEFINITION MunkTop.14.9: 3-ary function `Cplusray`. If $\texttt{SSORD}[R, X] \wedge a \in X$ then $\texttt{Cplusray}(a, X, R) \simeq \{x \in X : aRx \vee x = a\}$.

$\texttt{Cplusray}(a, X, R) \simeq (\iota z_0)(\texttt{SSORD}[R, X] \wedge a \in X \wedge z_0 \simeq (\iota y_0)(\forall x_0)(x_0 \in y_0 \leftrightarrow (\exists x)(x_0 = x \wedge (\varpi_0(a, x) \in R \vee x = a) \wedge (x \in X))))$

`DEFINITION MunkTop.14.10: 3-ary function Cminusray. If SSORD[R,X] \wedge a \in X then Cminusray(a,X,R) \simeq {x \in X : x \infixrl{R} a \vee x = a}.`

DEFINITION MunkTop.14.10: 3-ary function `Cminusray`. If $\texttt{SSORD}[R, X] \wedge a \in X$ then $\texttt{Cminusray}(a, X, R) \simeq \{x \in X : xRa \vee x = a\}$.

$\texttt{Cminusray}(a, X, R) \simeq (\iota z_0)(\texttt{SSORD}[R, X] \wedge a \in X \wedge z_0 \simeq (\iota y_0)(\forall x_0)(x_0 \in y_0 \leftrightarrow (\exists x)(x_0 = x \wedge (\varpi_0(x, a) \in R \vee x = a) \wedge (x \in X))))$

G.4 Section 15

```
DEFINITION MunkTop.15.1: 4-ary function Prodtopbasis.
If TOPSP[X,\mathscr{T}] \wedge
TOPSP[Y,\mathscr{T}'] then Prodtopbasis(X,\mathscr{T},Y,\mathscr{T}')
\simeq {U \times V : U \in \mathscr{T} \wedge V \in \mathscr{T}'}.
```

DEFINITION MunkTop.15.1: 4-ary function `Prodtopbasis`. If $\texttt{TOPSP}[X, \mathscr{T}] \wedge \texttt{TOPSP}[Y, \mathscr{T}']$ then $\texttt{Prodtopbasis}(X, \mathscr{T}, Y, \mathscr{T}') \simeq \{U \times V : U \in \mathscr{T} \wedge V \in \mathscr{T}'\}$.

$\texttt{Prodtopbasis}(X, \mathscr{T}, Y, \mathscr{T}') \simeq (\iota z_0)(\texttt{TOPSP}[X, \mathscr{T}] \wedge \texttt{TOPSP}[Y, \mathscr{T}'] \wedge z_0 \simeq (\iota y_0)(\forall x_0)(x_0 \in y_0 \leftrightarrow (\exists U, V)(x_0 = \times(U, V) \wedge (U \in \mathscr{T} \wedge V \in \mathscr{T}'))))$

```
DEFINITION MunkTop.15.2: 4-ary function Prodtop. If TOPSP[X,\mathscr{T}]
\wedge TOPSP[Y,\mathscr{T}'] then Prodtop(X,\mathscr{T},Y,\mathscr{T}')
\simeq Basisgentop(Prodtopbasis(X,\mathscr{T},Y,\mathscr{T}'),X).
```

DEFINITION MunkTop.15.2: 4-ary function `Prodtop`. If $\texttt{TOPSP}[X, \mathscr{T}] \wedge \texttt{TOPSP}[Y, \mathscr{T}']$ then $\texttt{Prodtop}(X, \mathscr{T}, Y, \mathscr{T}') \simeq \texttt{Basisgentop}(\texttt{Prodtopbasis}(X, \mathscr{T}, Y, \mathscr{T}'), X)$.

$\texttt{Prodtop}(X, \mathscr{T}, Y, \mathscr{T}') \simeq (\iota x_0)(\texttt{TOPSP}[X, \mathscr{T}] \wedge \texttt{TOPSP}[Y, \mathscr{T}'] \wedge x_0 \simeq \texttt{Basisgentop}(\texttt{Prodtopbasis}(X, \mathscr{T}, Y, \mathscr{T}'), X))$

```
DEFINITION MunkTop.15.3: 2-ary function \pi_{1}. \pi_{1}(x,y) \simeq x.
```

DEFINITION MunkTop.15.3: 2-ary function π_1. $\pi_1(x, y) \simeq x$.

$\pi_1(x, y) \simeq x$

```
DEFINITION MunkTop.15.4: 2-ary function \pi_{2}. \pi_{2}(x,y) \simeq y.
```

DEFINITION MunkTop.15.4: 2-ary function π_2. $\pi_2(x, y) \simeq y$.

$\pi_2(x, y) \simeq y$

```
DEFINITION MunkTop.15.5: 1-ary function \pi_{1}. If (\exists u,v)
(x = <u,v>) then \pi_{1}(x) \simeq u. Otherwise \pi_{1}(x) \up.
```

DEFINITION MunkTop.15.5: 1-ary function π_1. If $(\exists u, v)(x = \langle u, v\rangle)$ then $\pi_1(x) \simeq u$. Otherwise $\pi_1(x)\uparrow$.

$\pi_1(x) \simeq (\iota x_0)(((\exists u, v)x = \varpi_0(u, v) \wedge x_0 \simeq u))$

```
DEFINITION MunkTop.15.6: 1-ary function \pi_{2}. If (\exists u,v)
(x = <u,v>) then \pi_{2}(x) \simeq v. Otherwise \pi_{2}(x) \up.
```

DEFINITION MunkTop.15.6: 1-ary function π_2. If $(\exists u, v)(x = \langle u, v\rangle)$ then $\pi_2(x) \simeq v$. Otherwise $\pi_2(x)\uparrow$.

$\pi_2(x) \simeq (\iota x_0)(((\exists u, v)x = \varpi_0(u, v) \wedge x_0 \simeq v))$

G.5 Section 16

```
DEFINITION MunkTop.16.1: 3-ary function Subspacetop.
If TOPSP[X,\mathscr{T}] \wedge Y \subseteq X then
Subspacetop(Y,X,\mathscr{T}) \simeq {Y \cap U : U \in \mathscr{T}}.
```

DEFINITION MunkTop.16.1: 3-ary function `Subspacetop`. If $\texttt{TOPSP}[X, \mathscr{T}] \wedge Y \subseteq X$ then $\texttt{Subspacetop}(Y, X, \mathscr{T}) \simeq \{Y \cap U : U \in \mathscr{T}\}$.

$\texttt{Subspacetop}(Y, X, \mathscr{T}) \simeq (\iota z_0)(\texttt{TOPSP}[X, \mathscr{T}] \wedge \subseteq[Y, X] \wedge z_0 \simeq (\iota y_0)(\forall x_0)(x_0 \in y_0 \leftrightarrow (\exists Y, U)(x_0 = \cap(Y, U) \wedge (U \in \mathscr{T}))))$

```
DEFINITION MunkTop.16.2: 4-ary relation SUBSPACE.
If TOPSP[X,\mathscr{T}] \wedge Y \subseteq X then
SUBSPACE[Y,\mathscr{T}',X,\mathscr{T}] \iff
\mathscr{T}' = Subspacetop(Y,X,\mathscr{T}).
```

DEFINITION MunkTop.16.2: 4-ary relation SUBSPACE. If $\texttt{TOPSP}[X, \mathscr{T}] \wedge Y \subseteq X$ then $\texttt{SUBSPACE}[Y, \mathscr{T}', X, \mathscr{T}] \leftrightarrow \mathscr{T}' = \texttt{Subspacetop}(Y, X, \mathscr{T})$.

$\texttt{SUBSPACE}[Y, \mathscr{T}', X, \mathscr{T}] \leftrightarrow (\texttt{TOPSP}[X, \mathscr{T}] \wedge \subseteq[Y, X] \wedge \mathscr{T}' = \texttt{Subspacetop}(Y, X, \mathscr{T}))$

```
DEFINITION MunkTop.16.3: 4-ary function Dictionaryorder. If SSORD[R,A]
\wedge SSORD[S,B] then Dictionaryorder(R,A,S,B) \simeq (!T)(
  (\forall y)(
    y \in T \iff (\exists p,q,r,s)(
      y = <<p,q>,<r,s>> \wedge p,r \in A \wedge q,s \in B \wedge
      ( p \infixrl{R} r \vee p = r \wedge q \infixrl{S} s )
    )
  )
).
```

DEFINITION MunkTop.16.3: 4-ary function Dictionaryorder. If $\texttt{SSORD}[R, A] \wedge \texttt{SSORD}[S, B]$ then $\texttt{Dictionaryorder}(R, A, S, B) \simeq (!T)((\forall y)(y \in T \leftrightarrow (\exists p, q, r, s)(y = \langle \langle p, q \rangle, \langle r, s \rangle \rangle \wedge p, r \in A \wedge q, s \in B \wedge (pRr \vee p = r \wedge qSs))))$.

$\texttt{Dictionaryorder}(R, A, S, B) \simeq (\iota y_0)(\texttt{SSORD}[R, A] \wedge \texttt{SSORD}[S, B] \wedge y_0 \simeq (\iota x_0)(\exists T)(x_0 = T \wedge ((\forall y) y \in T \leftrightarrow (\exists p, q, r, s) y = \varpi_0(\varpi_0(p, q), \varpi_0(r, s)) \wedge p \in A \wedge r \in A \wedge q \in B \wedge s \in B \wedge \varpi_0(p, r) \in R \vee p = r \wedge \varpi_0(q, s) \in S)))$

```
DEFINITION MunkTop.16.4: 0-ary function Orderedsquare.
If I = Ccinterval(0_{R},1_{R},\mathbb{R},{<x,y> : x <_{R} y}) \wedge
R = Dictionaryorder({<x,y> : x <_{R} y},I,{<x,y> : x <_{R} y},I) then
Orderedsquare \simeq <I \times I, Ordertop(I \times I,R)>.
```

DEFINITION MunkTop.16.4: 0-ary function Orderedsquare.

If $I = \texttt{Ccinterval}(\texttt{0}_{\texttt{R}}, \texttt{1}_{\texttt{R}}, \mathbb{R}, \{\langle x, y \rangle : x <_{\texttt{R}} y\}) \wedge R = \texttt{Dictionaryorder}(\{\langle x, y \rangle : x <_{\texttt{R}} y\}, I, \{\langle x, y \rangle : x <_{\texttt{R}} y\}, I)$ then $\texttt{Orderedsquare} \simeq \langle I \times I, \texttt{Ordertop}(I \times I, R) \rangle$.

$\texttt{Orderedsquare} \simeq (\iota z_0)(I = \texttt{Ccinterval}(\texttt{0}_{\texttt{R}}, \texttt{1}_{\texttt{R}}, \mathbb{R}, (\iota y_0)(\forall x_0)(x_0 \in y_0 \leftrightarrow (\exists x, y)(x_0 = \varpi_0(x, y) \wedge (<_{\texttt{R}}[x, y])))) \wedge R = \texttt{Dictionaryorder}((\iota y_0)(\forall x_0)(x_0 \in y_0 \leftrightarrow (\exists x, y)(x_0 = \varpi_0(x, y) \wedge (<_{\texttt{R}}[x, y]))), I, (\iota y_0)(\forall x_0)(x_0 \in y_0 \leftrightarrow (\exists x, y)(x_0 = \varpi_0(x, y) \wedge (<_{\texttt{R}}[x, y]))), I) \wedge z_0 \simeq \varpi_0(\times(I, I), \texttt{Ordertop}(\times(I, I), R)))$

```
DEFINITION MunkTop.16.5: 3-ary relation CONVEX. If SSORD[R,X]
\wedge Y \subseteq X then CONVEX[Y,X,R] \iff
(\forall a,b \in Y)(
  Oointerval(a,b,X,R) \subseteq Y
).
```

DEFINITION MunkTop.16.5: 3-ary relation `CONVEX`. If $\texttt{SSORD}[R, X] \wedge Y \subseteq X$ then $\texttt{CONVEX}[Y, X, R] \leftrightarrow (\forall a, b \in Y)(\texttt{Oointerval}(a, b, X, R) \subseteq Y)$.

$\texttt{CONVEX}[Y, X, R] \leftrightarrow (\texttt{SSORD}[R, X] \wedge \subseteq[Y, X] \wedge (\forall a, b)(a \in Y \wedge b \in Y \rightarrow \subseteq[\texttt{Oointerval}(a, b, X, R), Y]))$

G.6 Section 17

```
DEFINITION MunkTop.17.1: 3-ary relation CLOSEDSET. If A \subseteq X then
CLOSEDSET[A,X,\mathscr{T}] \iff TOPSP[X,\mathscr{T}] \wedge
OPENSET[X \less A,X,\mathscr{T}].
```

DEFINITION MunkTop.17.1: 3-ary relation `CLOSEDSET`. If $A \subseteq X$ then $\texttt{CLOSEDSET}[A, X, \mathscr{T}] \leftrightarrow \texttt{TOPSP}[X, \mathscr{T}] \wedge \texttt{OPENSET}[X \backslash A, X, \mathscr{T}]$.

$\texttt{CLOSEDSET}[A, X, \mathscr{T}] \leftrightarrow (\subseteq[A, X] \wedge \texttt{TOPSP}[X, \mathscr{T}] \wedge \texttt{OPENSET}[\backslash(X, A), X, \mathscr{T}])$

```
DEFINITION MunkTop.17.2: 1-ary function Interior.
If TOPSP[X,\mathscr{T}] \wedge A \subseteq X then Interior(A) \simeq
\cup({U \in \mathscr{T} : U \subseteq A}).
```

DEFINITION MunkTop.17.2: 1-ary function `Interior`. If $\texttt{TOPSP}[X, \mathscr{T}] \wedge A \subseteq X$ then $\texttt{Interior}(A) \simeq \cup(\{U \in \mathscr{T} : U \subseteq A\})$.

$\texttt{Interior}(A) \simeq (\iota z_0)(\texttt{TOPSP}[X, \mathscr{T}] \wedge \subseteq[A, X] \wedge z_0 \simeq \cup((\iota y_0)(\forall x_0)(x_0 \in y_0 \leftrightarrow (\exists U)(x_0 = U \wedge (\subseteq[U, A]) \wedge (U \in \mathscr{T})))))$

```
DEFINITION MunkTop.17.3: 3-ary function Closure. If
TOPSP[X,\mathscr{T}] \wedge A \subseteq X then Closure(A,X,T) \simeq
\cap({F : A \subseteq F \wedge CLOSEDSET[F,X,\mathscr{T}]}).
```

DEFINITION MunkTop.17.3: 3-ary function `Closure`. If `TOPSP`$[X, \mathscr{T}] \wedge A \subseteq X$ then `Closure`$(A, X, T) \simeq \cap(\{F : A \subseteq F \wedge$ `CLOSEDSET`$[F, X, \mathscr{T}]\})$.

`Closure`$(A, X, T) \simeq (\iota z_0)($`TOPSP`$[X, \mathscr{T}] \wedge \subseteq [A, X] \wedge z_0 \simeq \cap((\iota y_0)(\forall x_0)(x_0 \in y_0 \leftrightarrow (\exists F)(x_0 = F \wedge (\subseteq [A, F] \wedge$ `CLOSEDSET`$[F, X, \mathscr{T}])))))$

```
DEFINITION MunkTop.17.4: 2-ary relation INTERSECTS. INTERSECTS[A,B] \iff
A \cap B \neq \varnothing.
```

DEFINITION MunkTop.17.4: 2-ary relation `INTERSECTS`. `INTERSECTS`$[A, B] \leftrightarrow A \cap B \neq \varnothing$.

`INTERSECTS`$[A, B] \leftrightarrow \neg(\cap(A, B) = \varnothing)$

```
DEFINITION MunkTop.17.4.5: 2-ary relation DISJOINT. DISJOINT[A,B] \iff
A \cap B = \varnothing.
```

DEFINITION MunkTop.17.4.5: 2-ary relation `DISJOINT`. `DISJOINT`$[A, B] \leftrightarrow A \cap B = \varnothing$.

`DISJOINT`$[A, B] \leftrightarrow \cap(A, B) = \varnothing$

```
DEFINITION MunkTop.17.5: 4-ary relation NBHD. If TOPSP[X,\mathscr{T}]
\wedge x \in X \wedge U \in \mathscr{T} then
NBHD[U,x,X,\mathscr{T}] \iff x \in U.
```

DEFINITION MunkTop.17.5: 4-ary relation `NBHD`. If `TOPSP`$[X, \mathscr{T}] \wedge x \in X \wedge U \in \mathscr{T}$ then `NBHD`$[U, x, X, \mathscr{T}] \leftrightarrow x \in U$.

`NBHD`$[U, x, X, \mathscr{T}] \leftrightarrow ($`TOPSP`$[X, \mathscr{T}] \wedge x \in X \wedge U \in \mathscr{T} \wedge x \in U)$

```
DEFINITION MunkTop.17.6: 4-ary relation LIMITPT.
If TOPSP[X,\mathscr{T}] \wedge x \in X \wedge A \subseteq X then
LIMITPT[x,A,X,\mathscr{T}] \iff (\forall U)(
  NBHD[U,x,X,\mathscr{T}] \implies INTERSECTS[U,A \less {x}]
).
```

DEFINITION MunkTop.17.6: 4-ary relation LIMITPT. If $\texttt{TOPSP}[X, \mathscr{T}] \wedge x \in X \wedge A \subseteq X$ then $\texttt{LIMITPT}[x, A, X, \mathscr{T}] \leftrightarrow (\forall U)(\texttt{NBHD}[U, x, X, \mathscr{T}] \rightarrow \texttt{INTERSECTS}[U, A \backslash \{x\}])$.

$\texttt{LIMITPT}[x, A, X, \mathscr{T}] \leftrightarrow (\texttt{TOPSP}[X, \mathscr{T}] \wedge x \in X \wedge \subseteq [A, X] \wedge (\forall U)\, \texttt{NBHD}[U, x, X, \mathscr{T}] \rightarrow \texttt{INTERSECTS}[U, \backslash (A, (\iota x_0)(\forall y_0)(y_0 \in x_0 \leftrightarrow y_0 = x))])$

```
DEFINITION MunkTop.17.7: 3-ary relation TOPCONV.
If TOPSP[X,\mathscr{T}] \wedge f \in Maps(\mathbb{N},X) \wedge
x \in X then TOPCONV[f,x,\mathscr{T}] \iff
(\forall U \in \mathscr{T})(
  NBHD[U,x,X,\mathscr{T}] \implies
  (\exists N \in \mathbb{N})(\forall n >_{N} N)(f(n) \in U)
).
```

DEFINITION MunkTop.17.7: 3-ary relation TOPCONV. If $\texttt{TOPSP}[X, \mathscr{T}] \wedge f \in \texttt{Maps}(\mathbb{N}, X) \wedge x \in X$ then $\texttt{TOPCONV}[f, x, \mathscr{T}] \leftrightarrow (\forall U \in \mathscr{T})(\texttt{NBHD}[U, x, X, \mathscr{T}] \rightarrow (\exists N \in \mathbb{N})(\forall n >_{\texttt{N}} N)(f(n) \in U))$.

$\texttt{TOPCONV}[f, x, \mathscr{T}] \leftrightarrow (\texttt{TOPSP}[X, \mathscr{T}] \wedge f \in \texttt{Maps}(\mathbb{N}, X) \wedge x \in X \wedge (\forall U)(U \in \mathscr{T} \rightarrow \texttt{NBHD}[U, x, X, \mathscr{T}] \rightarrow (\exists N)(N \in \mathbb{N} \wedge (\forall n)(>_{\texttt{N}}[n, N] \rightarrow (\iota x_0)(\varpi_0(n, x_0) \in f) \in U))))$

```
DEFINITION MunkTop.17.8: 2-ary relation HAUSDORFF.
If TOPSP[X,\mathscr{T}] then HAUSDORFF[X,\mathscr{T}] \iff
(\forall x,y \in X)(
  x \neq y \implies
  (\exists U,V \in \mathscr{T})(
    x \in U \wedge y \in V \wedge DISJOINT[U,V]
  )
).
```

DEFINITION MunkTop.17.8: 2-ary relation HAUSDORFF. If TOPSP$[X, \mathscr{T}]$ then HAUSDORFF$[X, \mathscr{T}] \leftrightarrow (\forall x, y \in X)(x \neq y \rightarrow (\exists U, V \in \mathscr{T})(x \in U \wedge y \in V \wedge \texttt{DISJOINT}[U, V]))$.

HAUSDORFF$[X, \mathscr{T}] \leftrightarrow (\texttt{TOPSP}[X, \mathscr{T}] \wedge (\forall x, y)(x \in X \wedge y \in X \rightarrow \neg(x = y) \rightarrow (\exists U, V)(U \in \mathscr{T} \wedge V \in \mathscr{T} \wedge x \in U \wedge y \in V \wedge \texttt{DISJOINT}[U, V])))$

```
DEFINITION MunkTop.17.9: 3-ary relation TOPLIMIT.
If HAUSDORFF[X,\mathscr{T}] \wedge
(\exists y)(TOPCONV[f,y,\mathscr{T}]) then
TOPLIMIT[x,f,\mathscr{T}] \iff
x = (!y)(TOPCONV[f,y,\mathscr{T}]).
```

DEFINITION MunkTop.17.9: 3-ary relation TOPLIMIT. If HAUSDORFF$[X, \mathscr{T}] \wedge (\exists y)(\texttt{TOPCONV}[f, y, \mathscr{T}])$ then TOPLIMIT$[x, f, \mathscr{T}] \leftrightarrow x = (!y)(\texttt{TOPCONV}[f, y, \mathscr{T}])$.

TOPLIMIT$[x, f, \mathscr{T}] \leftrightarrow (\texttt{HAUSDORFF}[X, \mathscr{T}] \wedge (\exists y)\, \texttt{TOPCONV}[f, y, \mathscr{T}] \wedge x = (\iota x_0)(\exists y)(x_0 = y \wedge (\texttt{TOPCONV}[f, y, \mathscr{T}])))$

G.7 Section 18

```
DEFINITION MunkTop.18.1: 5-ary relation CONTINUOUS. If TOPSP[X,T]
\wedge TOPSP[Y,T'] \wedge f \in Maps(X,Y) then
CONTINUOUS[f,X,T,Y,T'] \iff
(\forall V \in T')(Cnv(f) |_{Rng }V \in T).
```

DEFINITION MunkTop.18.1: 5-ary relation CONTINUOUS. If TOPSP$[X, T] \wedge$TOPSP$[Y, T'] \wedge f \in \texttt{Maps}(X, Y)$ then CONTINUOUS$[f, X, T, Y, T'] \leftrightarrow (\forall V \in T')(\texttt{Cnv}(f)|_{Rng} V \in T)$.

CONTINUOUS$[f, X, T, Y, T'] \leftrightarrow (\texttt{TOPSP}[X, T] \wedge \texttt{TOPSP}[Y, T'] \wedge f \in \texttt{Maps}(X, Y) \wedge (\forall V)(V \in T' \rightarrow |_{\texttt{Rng}}(\texttt{Cnv}(f), V) \in T))$

```
DEFINITION MunkTop.18.2: 6-ary relation CONTAT. If TOPSP[X,T] \wedge
TOPSP[Y,T'] \wedge f \in Maps(X,Y) \wedge x \in X then
CONTAT[x,f,X,T,Y,T'] \iff
(\forall V)(
  NBHD[V,f(x),Y,T'] \implies (\exists U)(
    NBHD[U,x,X,T] \wedge f |_{Rng} U \subseteq V
  )
).
```

DEFINITION MunkTop.18.2: 6-ary relation CONTAT. If $\texttt{TOPSP}[X, T] \wedge \texttt{TOPSP}[Y, T'] \wedge f \in \texttt{Maps}(X, Y) \wedge x \in X$ then $\texttt{CONTAT}[x, f, X, T, Y, T'] \leftrightarrow (\forall V)(\texttt{NBHD}[V, f(x), Y, T'] \rightarrow (\exists U)(\texttt{NBHD}[U, x, X, T] \wedge f|_{Rng} U \subseteq V))$.

$\texttt{CONTAT}[x, f, X, T, Y, T'] \leftrightarrow (\texttt{TOPSP}[X, T] \wedge \texttt{TOPSP}[Y, T'] \wedge f \in \texttt{Maps}(X, Y) \wedge x \in X \wedge (\forall V)\, \texttt{NBHD}[V, (\iota x_0)(\varpi_0(x, x_0) \in f), Y, T'] \rightarrow (\exists U)\, \texttt{NBHD}[U, x, X, T] \wedge \subseteq[|_{\texttt{Rng}}(f, U), V])$

```
DEFINITION MunkTop.18.3: 5-ary relation HOMEOMORPHISM. If TOPSP[X,T]
\wedge TOPSP[Y,T'] \wedge BIJ[f,X,Y] then
HOMEOMORPHISM[f,X,T,Y,T'] \iff
CONTINUOUS[f,X,T,Y,T'] \wedge CONTINUOUS[Cnv(f),Y,T',X,T].
```

DEFINITION MunkTop.18.3: 5-ary relation HOMEOMORPHISM. If $\texttt{TOPSP}[X, T] \wedge \texttt{TOPSP}[Y, T'] \wedge \texttt{BIJ}[f, X, Y]$ then $\texttt{HOMEOMORPHISM}[f, X, T, Y, T'] \leftrightarrow \texttt{CONTINUOUS}[f, X, T, Y, T'] \wedge \texttt{CONTINUOUS}[\texttt{Cnv}(f), Y, T', X, T]$.

$\texttt{HOMEOMORPHISM}[f, X, T, Y, T'] \leftrightarrow (\texttt{TOPSP}[X, T] \wedge \texttt{TOPSP}[Y, T'] \wedge \texttt{BIJ}[f, X, Y] \wedge \texttt{CONTINUOUS}[f, X, T, Y, T'] \wedge \texttt{CONTINUOUS}[\texttt{Cnv}(f), Y, T', X, T])$

```
DEFINITION MunkTop.18.3.5: 4-ary relation HOMEOMORPHIC. If TOPSP[X,T]
\wedge TOPSP[Y,T'] then HOMEOMORPHIC[X,T,Y,T'] \iff (\exists f)
(HOMEOMORPHISM[f,X,T,Y,T']).
```

DEFINITION MunkTop.18.3.5: 4-ary relation HOMEOMORPHIC. If $\texttt{TOPSP}[X, T] \wedge \texttt{TOPSP}[Y, T']$ then $\texttt{HOMEOMORPHIC}[X, T, Y, T'] \leftrightarrow (\exists f)(\texttt{HOMEOMORPHISM}[f, X, T, Y, T'])$.

$\texttt{HOMEOMORPHIC}[X, T, Y, T'] \leftrightarrow (\texttt{TOPSP}[X, T] \wedge \texttt{TOPSP}[Y, T'] \wedge (\exists f)\, \texttt{HOMEOMORPHISM}[f, X, T, Y, T'])$

```
DEFINITION MunkTop.18.4: 5-ary relation TOPIMBED. If TOPSP[X,T] \wedge
TOPSP[Y,T'] \wedge MONO[f,X,Y] \wedge CONTINUOUS[f,X,T,Y,T'] then
TOPIMBED[f,X,T,Y,T'] \iff
HOMEO[f,X,T,Rng(f),Subspacetop(Rng(f),Y,T')].
```

DEFINITION MunkTop.18.4: 5-ary relation TOPIMBED. If $\texttt{TOPSP}[X,T] \wedge \texttt{TOPSP}[Y,T'] \wedge \texttt{MONO}[f,X,Y] \wedge \texttt{CONTINUOUS}[f,X,T,Y,T']$ then $\texttt{TOPIMBED}[f,X,T,Y,T'] \leftrightarrow \texttt{HOMEO}[f,X,T,\texttt{Rng}(f),\texttt{Subspacetop}(\texttt{Rng}(f),Y,T')]$.

$\texttt{TOPIMBED}[f,X,T,Y,T'] \leftrightarrow (\texttt{TOPSP}[X,T] \wedge \texttt{TOPSP}[Y,T'] \wedge \texttt{MONO}[f,X,Y] \wedge \texttt{CONTINUOUS}[f,X,T,Y,T'] \wedge \texttt{HOMEOMORPHISM}[f,X,T,\texttt{Rng}(f),\texttt{Subspacetop}(\texttt{Rng}(f),Y,T')])$

G.8 Section 19

```
DEFINITION MunkTop.19.1: 3-ary relation TUPLE. If FCN[f] \wedge
Dom(f) = J then TUPLE[x,f,J] \iff
FCN[x] \wedge Dom(x) = J \wedge
(\forall \alpha \in J)(x(\alpha) \in f(\alpha)).
```

DEFINITION MunkTop.19.1: 3-ary relation TUPLE. If $\texttt{FCN}[f] \wedge \texttt{Dom}(f) = J$ then $\texttt{TUPLE}[x,f,J] \leftrightarrow \texttt{FCN}[x] \wedge \texttt{Dom}(x) = J \wedge (\forall \alpha \in J)(x(\alpha) \in f(\alpha))$.

$\texttt{TUPLE}[x,f,J] \leftrightarrow (\texttt{FCN}[f] \wedge \texttt{Dom}(f) = J \wedge \texttt{FCN}[x] \wedge \texttt{Dom}(x) = J \wedge (\forall \alpha)(\alpha \in J \rightarrow (\iota x_0)(\varpi_0(\alpha, x_0) \in x) \in (\iota x_0)(\varpi_0(\alpha, x_0) \in f)))$

```
DEFINITION MunkTop.19.2: 2-ary function Cartesprod. If FCN[f]
\wedge Dom(f) = J then Cartesprod(f,J) \simeq {x: TUPLE[x,f,J]}.
```

DEFINITION MunkTop.19.2: 2-ary function Cartesprod. If $\texttt{FCN}[f] \wedge \texttt{Dom}(f) = J$ then $\texttt{Cartesprod}(f,J) \simeq \{x : \texttt{TUPLE}[x,f,J]\}$.

$\texttt{Cartesprod}(f,J) \simeq (\iota z_0)(\texttt{FCN}[f] \wedge \texttt{Dom}(f) = J \wedge z_0 \simeq (\iota y_0)(\forall x_0)(x_0 \in y_0 \leftrightarrow (\exists x)(x_0 = x \wedge (\texttt{TUPLE}[x,f,J]))))$

```
DEFINITION MunkTop.19.2.5: 2-ary function Cartespow. Cartespow(A,B)
\simeq Cartesprod((\lambda b \in B)(A),B).
```

DEFINITION MunkTop.19.2.5: 2-ary function Cartespow. Cartespow$(A, B) \simeq$ Cartesprod$((\lambda b \in B)(A), B)$.

Cartespow$(A, B) \simeq$ Cartesprod$((\iota z_0)(\forall y_0)(y_0 \in z_0 \leftrightarrow (\exists b, x_0)(y_0 = \varpi_0(b, x_0) \wedge x_0 = (A) \wedge b \in B)), B)$

```
DEFINITION MunkTop.19.3: 3-ary function Boxtopbasis. If FCN[X] \wedge
Dom(X) = J \wedge FCN[T] \wedge Dom(T) = J \wedge
(\forall \alpha \in J)(TOPSP[X(\alpha),T(\alpha)]) then
Boxtopbasis(X,T,J) \simeq
{Cartesprod(U,J) :
  FCN[U] \wedge
  Dom(U) = J \wedge
  (\forall \alpha \in J)(U(\alpha) \in T(\alpha))
}.
```

DEFINITION MunkTop.19.3: 3-ary function Boxtopbasis. If FCN$[X] \wedge$ Dom$(X) = J \wedge$FCN$[T] \wedge$Dom$(T) = J \wedge (\forall \alpha \in J)($TOPSP$[X(\alpha), T(\alpha)])$ then Boxtopbasis$(X, T, J) \simeq \{$Cartesprod$(U, J) :$ FCN$[U] \wedge$ Dom$(U) = J \wedge (\forall \alpha \in J)(U(\alpha) \in T(\alpha))\}$.

Boxtopbasis$(X, T, J) \simeq (\iota x_1)(FCN[X] \wedge$Dom$(X) = J \wedge$FCN$[T] \wedge$Dom$(T) = J \wedge (\forall \alpha)(\alpha \in J \rightarrow$ TOPSP$[(\iota x_0)(\varpi_0(\alpha, x_0) \in X), (\iota x_0)(\varpi_0(\alpha, x_0) \in T)]) \wedge x_1 \simeq (\iota z_0)(\forall y_0)(y_0 \in z_0 \leftrightarrow (\exists U, J)(y_0 =$ Cartesprod$(U, J) \wedge ($FCN$[U] \wedge$ Dom$(U) = J \wedge (\forall \alpha)(\alpha \in J \rightarrow (\iota x_0)(\varpi_0(\alpha, x_0) \in U) \in (\iota x_0)(\varpi_0(\alpha, x_0) \in T))))))$

```
DEFINITION MunkTop.19.4: 3-ary function Boxtop. If FCN[X] \wedge
Dom(X) = J \wedge FCN[T] \wedge Dom(T) = J \wedge
(\forall \alpha \in J)(TOPSP[X(\alpha),T(\alpha)]) then
Boxtop(X,T,J) \simeq <
  Cartesprod(X,J),
  Basisgentop(Boxtopbasis(X,T,J),Cartesprod(X,J))
>.
```

DEFINITION MunkTop.19.4: 3-ary function `Boxtop`. If $\texttt{FCN}[X] \wedge \texttt{Dom}(X) = J \wedge \texttt{FCN}[T] \wedge \texttt{Dom}(T) = J \wedge (\forall \alpha \in J)(\texttt{TOPSP}[X(\alpha), T(\alpha)])$ then $\texttt{Boxtop}(X, T, J) \simeq \langle \texttt{Cartesprod}(X, J), \texttt{Basisgentop}(\texttt{Boxtopbasis}(X, T, J), \texttt{Cartesprod}(X, J)) \rangle$.

$\texttt{Boxtop}(X, T, J) \simeq (\iota y_0)(\texttt{FCN}[X] \wedge \texttt{Dom}(X) = J \wedge \texttt{FCN}[T] \wedge \texttt{Dom}(T) = J \wedge (\forall \alpha)(\alpha \in J \rightarrow \texttt{TOPSP}[(\iota x_0)(\varpi_0(\alpha, x_0) \in X), (\iota x_0)(\varpi_0(\alpha, x_0) \in T)]) \wedge y_0 \simeq \varpi_0(\texttt{Cartesprod}(X, J), \texttt{Basisgentop}(\texttt{Boxtopbasis}(X, T, J), \texttt{Cartesprod}(X, J))))$

```
DEFINITION MunkTop.19.5: 3-ary function Projection. If FCN[f]
\wedge Dom(f) = J \wedge \beta \in J then
Projection(\beta,f,J) \simeq
(\lambda x \in Cartesprod(f,J))(x(\beta)).
```

DEFINITION MunkTop.19.5: 3-ary function `Projection`. If $\texttt{FCN}[f] \wedge \texttt{Dom}(f) = J \wedge \beta \in J$ then $\texttt{Projection}(\beta, f, J) \simeq (\lambda x \in \texttt{Cartesprod}(f, J))(x(\beta))$.

$\texttt{Projection}(\beta, f, J) \simeq (\iota y_1)(\texttt{FCN}[f] \wedge \texttt{Dom}(f) = J \wedge \beta \in J \wedge y_1 \simeq (\iota x_1)(\forall z_0)(z_0 \in x_1 \leftrightarrow (\exists x, y_0)(z_0 = \varpi_0(x, y_0) \wedge y_0 = ((\iota x_0)(\varpi_0(\beta, x_0) \in x)) \wedge x \in \texttt{Cartesprod}(f, J))))$

```
DEFINITION MunkTop.19.6: 3-ary function Prodtopsubbasis. If FCN[X]
\wedge Dom(X) = J \wedge FCN[T] \wedge Dom(T) = J \wedge
(\forall \alpha \in J)(TOPSP[X(\alpha),T(\alpha)]) then
Prodtopsubbasis(X,T,J) \simeq {
  Cnv(Projection(\beta,X,J)) |_{Rng} U : \beta \in J \wedge U \in T(\beta)
}.
```

DEFINITION MunkTop.19.6: 3-ary function `Prodtopsubbasis`. If $\texttt{FCN}[X] \wedge \texttt{Dom}(X) = J \wedge \texttt{FCN}[T] \wedge \texttt{Dom}(T) = J \wedge (\forall \alpha \in J)(\texttt{TOPSP}[X(\alpha), T(\alpha)])$ then $\texttt{Prodtopsubbasis}(X, T, J) \simeq \{\texttt{Cnv}(\texttt{Projection}(\beta, X, J))|_{Rng} U : \beta \in J \wedge U \in T(\beta)\}$.

$\texttt{Prodtopsubbasis}(X, T, J) \simeq (\iota x_1)(\texttt{FCN}[X] \wedge \texttt{Dom}(X) = J \wedge \texttt{FCN}[T] \wedge \texttt{Dom}(T) = J \wedge (\forall \alpha)(\alpha \in J \rightarrow \texttt{TOPSP}[(\iota x_0)(\varpi_0(\alpha, x_0) \in X), (\iota x_0)(\varpi_0(\alpha, x_0) \in T)]) \wedge x_1 \simeq (\iota z_0)(\forall y_0)(y_0 \in z_0 \leftrightarrow (\exists \beta, X, J, U)(y_0 = |_{\texttt{Rng}}(\texttt{Cnv}(\texttt{Projection}(\beta, X, J)), U) \wedge (\beta \in J \wedge U \in (\iota x_0)(\varpi_0(\beta, x_0) \in T)))))$

```
DEFINITION MunkTop.19.7: 3-ary function Productspace. If FCN[X] \wedge
Dom(X) = J \wedge FCN[T] \wedge Dom(T) = J \wedge
(\forall \alpha \in J)(TOPSP[X(\alpha),T(\alpha)]) then
Productspace(X,T,J) \simeq <
Cartesprod(X,J),
Subbasisgentop(Prodtopsubbasis(X,T,J),Cartesprod(X,J))
>.
```

DEFINITION MunkTop.19.7: 3-ary function Productspace. If $\texttt{FCN}[X] \wedge \texttt{Dom}(X) = J \wedge \texttt{FCN}[T] \wedge \texttt{Dom}(T) = J \wedge (\forall \alpha \in J)(\texttt{TOPSP}[X(\alpha), T(\alpha)])$ then $\texttt{Productspace}(X, T, J) \simeq \langle \texttt{Cartesprod}(X, J), \texttt{Subbasisgentop}(\texttt{Prodtopsubbasis}(X, T, J), \texttt{Cartesprod}(X, J)) \rangle$.

$\texttt{Productspace}(X, T, J) \simeq (\iota y_0)(\texttt{FCN}[X] \wedge \texttt{Dom}(X) = J \wedge \texttt{FCN}[T] \wedge \texttt{Dom}(T) = J \wedge (\forall \alpha)(\alpha \in J \rightarrow \texttt{TOPSP}[(\iota x_0)(\varpi_0(\alpha, x_0) \in X), (\iota x_0)(\varpi_0(\alpha, x_0) \in T)]) \wedge y_0 \simeq \varpi_0(\texttt{Cartesprod}(X, J), \texttt{Subbasisgentop}(\texttt{Prodtopsubbasis}(X, T, J), \texttt{Cartesprod}(X, J))))$

G.9 Section 20

```
DEFINITION MunkTop.20.1: 2-ary relation METRIC. METRIC[d,X] \iff
d \in Maps(X \times X,\mathbb{R}) \wedge
(\forall x,y \in X)(d(x,y) \geq_{R} 0_{R} \wedge
(d(x,y) = 0_{R} \iff x = y)) \wedge
(\forall x,y \in X)(d(x,y) = d(y,x)) \wedge
(\forall x,y,z \in X)(d(x,y) +_{R} d(y,z) \geq_{R} d(x,z)).
```

DEFINITION MunkTop.20.1: 2-ary relation METRIC. $\texttt{METRIC}[d, X] \leftrightarrow d \in \texttt{Maps}(X \times X, \mathbb{R}) \wedge (\forall x, y \in X)(d(x, y) \geq_{\texttt{R}} 0_{\texttt{R}} \wedge (d(x, y) = 0_{\texttt{R}} \leftrightarrow x = y)) \wedge (\forall x, y \in X)(d(x, y) = d(y, x)) \wedge (\forall x, y, z \in X)(d(x, y) +_R d(y, z) \geq_{\texttt{R}} d(x, z))$.

$\texttt{METRIC}[d, X] \leftrightarrow d \in \texttt{Maps}(\times(X, X), \mathbb{R}) \wedge (\forall x, y)(x \in X \wedge y \in X \rightarrow \geq_{\texttt{R}}[(\iota x_0)(\varpi_0(\varpi_0(x, y), x_0) \in d), 0_{\texttt{R}}] \wedge (\iota x_0)(\varpi_0(\varpi_0(x, y), x_0) \in d) = 0_{\texttt{R}} \leftrightarrow x = y) \wedge (\forall x, y)(x \in X \wedge y \in X \rightarrow (\iota x_0)(\varpi_0(\varpi_0(x, y), x_0) \in d) = (\iota x_0)(\varpi_0(\varpi_0(y, x), x_0) \in d)) \wedge (\forall x, y, z)(x \in X \wedge y \in X \wedge z \in X \rightarrow \geq_{\texttt{R}}[+_{\texttt{R}}((\iota x_0)(\varpi_0(\varpi_0(x, y), x_0) \in d), (\iota x_0)(\varpi_0(\varpi_0(y, z), x_0) \in d)), (\iota x_0)(\varpi_0(\varpi_0(x, z), x_0) \in d)])$

```
DEFINITION MunkTop.20.2: 4-ary function Ball. If METRIC[d,X] \wedge
x \in X \wedge \veps >_{R} 0_{R} then Ball(\veps,x,d,X) \simeq
{y \in X : d(x,y) <_{R} \veps}.
```

DEFINITION MunkTop.20.2: 4-ary function `Ball`. If $\texttt{METRIC}[d, X] \wedge x \in X \wedge \varepsilon >_{\texttt{R}} 0_{\texttt{R}}$ then $\texttt{Ball}(\varepsilon, x, d, X) \simeq \{y \in X : d(x, y) <_{\texttt{R}} \varepsilon\}$.

$\texttt{Ball}(\varepsilon, x, d, X) \simeq (\iota x_1)(\texttt{METRIC}[d, X] \wedge x \in X \wedge >_{\texttt{R}}[\varepsilon, 0_{\texttt{R}}] \wedge x_1 \simeq (\iota z_0)(\forall y_0)(y_0 \in z_0 \leftrightarrow (\exists y)(y_0 = y \wedge (<_{\texttt{R}}[(\iota x_0)(\varpi_0(\varpi_0(x, y), x_0) \in d), \varepsilon]) \wedge (y \in X))))$

```
DEFINITION MunkTop.20.3: 2-ary function Metrictopbasis. If METRIC[d,X]
then Metrictopbasis(d,X) \simeq
{Ball(\veps,x,d,X) : x \in X \wedge \veps >_{R} 0_{R}}.
```

DEFINITION MunkTop.20.3: 2-ary function `Metrictopbasis`. If $\texttt{METRIC}[d, X]$ then $\texttt{Metrictopbasis}(d, X) \simeq \{\texttt{Ball}(\varepsilon, x, d, X) : x \in X \wedge \varepsilon >_{\texttt{R}} 0_{\texttt{R}}\}$.

$\texttt{Metrictopbasis}(d, X) \simeq (\iota z_0)(\texttt{METRIC}[d, X] \wedge z_0 \simeq (\iota y_0)(\forall x_0)$
$(x_0 \in y_0 \leftrightarrow (\exists \varepsilon, x, d, X)(x_0 = \texttt{Ball}(\varepsilon, x, d, X) \wedge (x \in X \wedge >_{\texttt{R}}[\varepsilon, 0_{\texttt{R}}]))))$

```
DEFINITION MunkTop.20.4: 2-ary function Metrictop. If METRIC[d,X] then
Metrictop(d,X) \simeq Basisgentop(Metrictopbasis(d,X),X).
```

DEFINITION MunkTop.20.4: 2-ary function `Metrictop`. If $\texttt{METRIC}[d, X]$ then $\texttt{Metrictop}(d, X) \simeq \texttt{Basisgentop}(\texttt{Metrictopbasis}(d, X), X)$.

$\texttt{Metrictop}(d, X) \simeq (\iota x_0)(\texttt{METRIC}[d, X] \wedge x_0 \simeq \texttt{Basisgentop}(\texttt{Metrictopbasis}(d, X), X))$

```
DEFINITION MunkTop.20.5: 2-ary relation METRIZABLE. If TOPSP[X,T] then
METRIZABLE[X,T] \iff (\exists d)(T = Metrictop(d,X)).
```

DEFINITION MunkTop.20.5: 2-ary relation `METRIZABLE`. If $\texttt{TOPSP}[X, T]$ then $\texttt{METRIZABLE}[X, T] \leftrightarrow (\exists d)(T = \texttt{Metrictop}(d, X))$.

$\texttt{METRIZABLE}[X, T] \leftrightarrow (\texttt{TOPSP}[X, T] \wedge (\exists d) T = \texttt{Metrictop}(d, X))$

```
DEFINITION MunkTop.20.6: 3-ary relation METRICSPACE. If TOPSP[X,T] then
METRICSPACE[X,T,d] \iff T = Metrictop(d,X).
```

DEFINITION MunkTop.20.6: 3-ary relation METRICSPACE. If TOPSP$[X, T]$ then METRICSPACE$[X, T, d] \leftrightarrow T =$ Metrictop(d, X).

METRICSPACE$[X, T, d] \leftrightarrow$ (TOPSP$[X, T] \wedge T =$ Metrictop(d, X))

```
DEFINITION MunkTop.20.7: 3-ary relation BOUNDED. If METRICSPACE[X,T,d]
\wedge A \subseteq X then BOUNDED[A,X,d] \iff
(\exists M \in \mathbb{R})
(\forall a_{1},a_{2} \in A)(d(a_{1},a_{2}) \leq_{R} M).
```

DEFINITION MunkTop.20.7: 3-ary relation BOUNDED. If METRICSPACE$[X, T, d] \wedge A \subseteq X$ then BOUNDED$[A, X, d] \leftrightarrow (\exists M \in \mathbb{R})(\forall a_1, a_2 \in A)(d(a_1, a_2) \leq_{\mathtt{R}} M)$.

BOUNDED$[A, X, d] \leftrightarrow$ (METRICSPACE$[X, T, d] \wedge \subseteq [A, X] \wedge (\exists M)(M \in \mathbb{R} \wedge (\forall a_1, a_2)(a_1 \in A \wedge a_2 \in A \rightarrow \leq_{\mathtt{R}}[(\iota x_0)(\varpi_0(\varpi_0(a_1, a_2), x_0) \in d), M]))))$

```
DEFINITION MunkTop.20.8: 1-ary function Diam. If METRICSPACE[X,T,d]
\wedge A \subseteq X \wedge A \neq \varnothing \wedge
BOUNDED[A,X,d] then Diam(A) \simeq (!s)(
  SUP[s,{<x,y>: x <_{R} y},{d(a_{1},a_{2}) : a_{1},a_{2} \in A}]
).
```

DEFINITION MunkTop.20.8: 1-ary function Diam. If METRICSPACE$[X, T, d] \wedge A \subseteq X \wedge A \neq \varnothing \wedge$ BOUNDED$[A, X, d]$ then Diam$(A) \simeq (!s)($SUP$[s, \{\langle x, y\rangle : x <_{\mathtt{R}} y\}, \{d(a_1, a_2) : a_1, a_2 \in A\}])$.

Diam$(A) \simeq (\iota y_1)($METRICSPACE$[X, T, d] \wedge \subseteq [A, X] \wedge \neg(A = \varnothing) \wedge$ BOUNDED$[A, X, d] \wedge y_1 \simeq (\iota x_1)(\exists s)(x_1 = s \wedge (SUP[s, (\iota y_0)(\forall x_0)(x_0 \in y_0 \leftrightarrow (\exists x, y)(x_0 = \varpi_0(x, y) \wedge (<_{\mathtt{R}}[x, y]))), (\iota z_0)(\forall y_0)(y_0 \in z_0 \leftrightarrow (\exists a_1, a_2, d)(y_0 = (\iota x_0)(\varpi_0(\varpi_0(a_1, a_2), x_0) \in d) \wedge (a_1 \in A \wedge a_2 \in A)))])))$

```
DEFINITION MunkTop.20.9: 2-ary function Stdbddmetric.
If METRICSPACE[X,T,d] then Stdbddmetric(d,X) \simeq (!d')(
  FCN[d',X \times X,\mathbb{R}] \wedge
  (\forall x,y \in X)(d'(x,y) = Min_{R}({d(x,y),1_{R}}))
).
```

DEFINITION MunkTop.20.9: 2-ary function `Stdbddmetric`. If $\texttt{METRICSPACE}[X, T, d]$ then $\texttt{Stdbddmetric}(d, X) \simeq (!d')(\texttt{FCN}[d', X \times X, \mathbb{R}] \wedge (\forall x, y \in X)$ $(d'(x, y) = \texttt{Min}_{\texttt{R}}(\{d(x, y), 1_{\texttt{R}}\})))$.

$\texttt{Stdbddmetric}(d, X) \simeq (\iota y_1)(\texttt{METRICSPACE}[X, T, d] \wedge y_1 \simeq (\iota x_1)(\exists d')$ $(x_1 = d' \wedge (\texttt{FCN}[d', \times(X, X), \mathbb{R}] \wedge (\forall x, y)(x \in X \wedge y \in X \to (\iota x_0)(\varpi_0(\varpi_0(x, y), x_0) \in$ $d') = \texttt{Min}_{\texttt{R}}((\iota y_0)(\forall z_0)(z_0 \in y_0 \leftrightarrow z_0 = (\iota x_0)(\varpi_0(\varpi_0(x, y), x_0) \in d) \vee z_0 = 1_{\texttt{R}}))))))$

```
DEFINITION MunkTop.20.10: 2-ary function Rnnorm. If n \in \omega \wedge
x \in Cartespow(\mathbb{R},n)
then Rnnorm(n,x) \simeq Sqrt_{R}(
  Finitesum_{R}(
    (\lambda m \in n)(x(m) \cdot_{R} x(m))
  )
).
```

DEFINITION MunkTop.20.10: 2-ary function `Rnnorm`. If $n \in \omega \wedge x \in \texttt{Cartespow}(\mathbb{R}, n)$ then $\texttt{Rnnorm}(n, x) \simeq \texttt{Sqrt}_{\texttt{R}}(\texttt{Finitesum}_{\texttt{R}}((\lambda m \in n)(x(m) \cdot_R x(m))))$.

$\texttt{Rnnorm}(n, x) \simeq (\iota y_1)(n \in \omega \wedge x \in \texttt{Cartespow}(\mathbb{R}, n) \wedge y_1 \simeq \texttt{Sqrt}_{\texttt{R}}(\texttt{Finitesum}_{\texttt{R}}((\iota x_1)(\forall z_0)(z_0 \in$ $x_1 \leftrightarrow (\exists m, y_0)(z_0 = \varpi_0(m, y_0) \wedge y_0 = (\cdot_{\texttt{R}}((\iota x_0)(\varpi_0(m, x_0) \in x), (\iota x_0)(\varpi_0(m, x_0) \in$ $x))) \wedge m \in n)))))$

```
DEFINITION MunkTop.20.11: 3-ary function Rneuclideanmetric.
If n \in \omega \wedge x,y \in Cartespow(\mathbb{R},n) then
Rneuclideanmetric(n,x,y) \simeq Rnnorm(n,x -_{R} y).
```

DEFINITION MunkTop.20.11: 3-ary function `Rneuclideanmetric`. If $n \in \omega \wedge x, y \in \texttt{Cartespow}(\mathbb{R}, n)$ then $\texttt{Rneuclideanmetric}(n, x, y) \simeq \texttt{Rnnorm}(n, x -_R y)$.

$\texttt{Rneuclideanmetric}(n, x, y) \simeq (\iota x_0)(n \in \omega \wedge x \in \texttt{Cartespow}(\mathbb{R}, n) \wedge y \in \texttt{Cartespow}(\mathbb{R}, n) \wedge x_0 \simeq \texttt{Rnnorm}(n, -_{\texttt{R}}(x, y)))$

```
DEFINITION MunkTop.20.12: 3-ary function Rnsqmetric. If n \in \omega \wedge
x,y \in Cartespow(\mathbb{R},n) then Rnsqmetric(n,x,y) \simeq
Max_{R}(
  {Av_{R}(x(i) -_{R} y(i)) : i \in n}
).
```

DEFINITION MunkTop.20.12: 3-ary function `Rnsqmetric`. If $n \in \omega \wedge x, y \in \texttt{Cartespow}(\mathbb{R}, n)$ then $\texttt{Rnsqmetric}(n, x, y) \simeq \texttt{Max}_{\texttt{R}}(\{\texttt{Av}_{\texttt{R}}(x(i) -_R y(i)) : i \in n\})$.

$\texttt{Rnsqmetric}(n, x, y) \simeq (\iota x_1)(n \in \omega \wedge x \in \texttt{Cartespow}(\mathbb{R}, n) \wedge y \in \texttt{Cartespow}(\mathbb{R}, n) \wedge x_1 \simeq \texttt{Max}_{\texttt{R}}((\iota z_0)(\forall y_0)(y_0 \in z_0 \leftrightarrow (\exists i, x, y)(y_0 = \texttt{Av}_{\texttt{R}}(-_{\texttt{R}}((\iota x_0)(\varpi_0(i, x_0) \in x), (\iota x_0)(\varpi_0(i, x_0) \in y))) \wedge (i \in n)))))$

```
DEFINITION MunkTop.20.13: 3-ary function Uniformmetric. If x,y \in
Cartespow(\mathbb{R},J) \wedge d' = Stdbddmetric(
  (\lambda u,v)(Rneuclideanmetric(1,u,v)),
  \mathbb{R}
)
then Uniformmetric(J,x,y) \simeq Sup_{R}(
  {d'(x(\alpha),y(\alpha)) : \alpha \in J}
).
```

DEFINITION MunkTop.20.13: 3-ary function `Uniformmetric`. If $x, y \in \texttt{Cartespow}(\mathbb{R}, J) \wedge d' = \texttt{Stdbddmetric}((\lambda u, v)(\texttt{Rneuclideanmetric}(1, u, v)), \mathbb{R})$ then $\texttt{Uniformmetric}(J, x, y) \simeq \texttt{Sup}_{\texttt{R}}(\{d'(x(\alpha), y(\alpha)) : \alpha \in J\})$.

$\texttt{Uniformmetric}(J, x, y) \simeq (\iota y_1)(x \in \texttt{Cartespow}(\mathbb{R}, J) \wedge y \in \texttt{Cartespow}(\mathbb{R}, J) \wedge d' = \texttt{Stdbddmetric}((\iota z_0)(\forall y_0)(y_0 \in z_0 \leftrightarrow (\exists u, v, x_0)(y_0 = \varpi_0(\varpi_0(u, v), x_0) \wedge x_0 = (\texttt{Rneuclideanmetric}(1_{\mathbb{N}}, u, v)))), \mathbb{R}) \wedge y_1 \simeq \texttt{Sup}_{\texttt{R}}((\iota x_1)(\forall z_0)(z_0 \in x_1 \leftrightarrow (\exists \alpha, x, y, d')(z_0 = (\iota y_0)(\varpi_0(\varpi_0((\iota x_0)(\varpi_0(\alpha, x_0) \in x), (\iota x_0)(\varpi_0(\alpha, x_0) \in y)), y_0) \in d') \wedge (\alpha \in J)))))$

G.10 Section 21

```
DEFINITION MunkTop.21.1: 3-ary relation CNTBLBASISAT. If TOPSP[X,T]
\wedge x \in X then CNTBLBASISAT[X,T,x] \iff
(\exists f : FCN[f,\mathbb{N},T])(
  (\forall U : NBHD[U,x,X,T])(
    (\exists n \in \mathbb{N})(
      f(n) \subseteq U
    )
  )
).
```

DEFINITION MunkTop.21.1: 3-ary relation CNTBLBASISAT. If TOPSP$[X, T] \wedge x \in X$ then CNTBLBASISAT$[X, T, x] \leftrightarrow (\exists f : \text{FCN}[f, \mathbb{N}, T])((\forall U : \text{NBHD}[U, x, X, T])((\exists n \in \mathbb{N})(f(n) \subseteq U)))$.

CNTBLBASISAT$[X, T, x] \leftrightarrow (\text{TOPSP}[X, T] \wedge x \in X \wedge (\exists f)(\text{FCN}[f, \mathbb{N}, T] \wedge (\forall U)(\text{NBHD}[U, x, X, T] \rightarrow (\exists n)(n \in \mathbb{N} \wedge \subseteq[(\iota x_0)(\varpi_0(n, x_0) \in f), U]))))$

```
DEFINITION MunkTop.21.2: 2-ary relation FIRSTCNTBL. If TOPSP[X,T]
then FIRSTCNTBL[X,T] \iff (\forall x \in X)(CNTBLBASISAT[X,T,x]).
```

DEFINITION MunkTop.21.2: 2-ary relation FIRSTCNTBL. If TOPSP$[X, T]$ then FIRSTCNTBL$[X, T] \leftrightarrow (\forall x \in X)(\text{CNTBLBASISAT}[X, T, x])$.

FIRSTCNTBL$[X, T] \leftrightarrow (\text{TOPSP}[X, T] \wedge (\forall x)(x \in X \rightarrow \text{CNTBLBASISAT}[X, T, x]))$

```
DEFINITION MunkTop.21.3: 7-ary relation UNIFCONV. If TOPSP[X,T] \wedge
METRICSPACE[Y,T',d] \wedge FCN[F] \wedge Dom(F) = \mathbb{N} \wedge
(\forall n \in \mathbb{N})(FCN[F(n),X,Y]) then
UNIFCONV[F,f,X,T,Y,T',d] \iff (\forall \veps \in \mathbb{R})(
  \veps >_{R} 0_{R} \implies (\exists N \in \mathbb{N})(
    (\forall n >_{N} N)(\forall x \in X)(
      d(F(n)(x),f(x)) <_{R} \veps
    )
  )
).
```

DEFINITION MunkTop.21.3: 7-ary relation UNIFCONV. If TOPSP$[X,T]\wedge$ METRICSPACE$[Y,T',d]\wedge$ FCN$[F]\wedge$ Dom$(F)=\mathbb{N}\wedge(\forall n\in\mathbb{N})(FCN[F(n),X,Y])$ then UNIFCONV$[F,f,X,T,Y,T',d]\leftrightarrow(\forall\varepsilon\in\mathbb{R})(\varepsilon>_{\mathtt{R}}0_{\mathtt{R}}\rightarrow(\exists N\in\mathbb{N})((\forall n>_{\mathtt{N}}N)(\forall x\in X)$ $(d(F(n)(x),f(x))<_{\mathtt{R}}\varepsilon)))$.

UNIFCONV$[F,f,X,T,Y,T',d]\leftrightarrow($TOPSP$[X,T]\wedge$METRICSPACE$[Y,T',d]\wedge$FCN$[F]\wedge$Dom$(F)=$ $\mathbb{N}\wedge(\forall n)(n\in\mathbb{N}\rightarrow$ FCN$[(\iota x_0)(\varpi_0(n,x_0)\in F),X,Y])\wedge(\forall\varepsilon)(\varepsilon\in\mathbb{R}\rightarrow>_{\mathtt{R}}[\varepsilon,0_{\mathtt{R}}]\rightarrow$ $(\exists N)(N\in\mathbb{N}\wedge(\forall n)(>_{\mathtt{N}}[n,N]\rightarrow(\forall x)(x\in X\rightarrow<_{\mathtt{R}}[(\iota z_0)(\varpi_0(\varpi_0((\iota y_0)(\varpi_0(x,y_0)\in$ $(\iota x_0)(\varpi_0(n,x_0)\in F)),(\iota x_0)(\varpi_0(x,x_0)\in f)),z_0)\in d),\varepsilon])))))$

G.11 Section 22

```
DEFINITION MunkTop.22.1: 5-ary relation QUOTIENTMAP. If TOPSP[X,T]
\wedge TOPSP[Y,T'] \wedge SURJ[p,X,Y] then
QUOTIENTMAP[p,X,T,Y,T'] \iff (\forall U \subseteq Y)(
  U \in T' \iff Cnv(p) |_{Rng} U \in T
).
```

DEFINITION MunkTop.22.1: 5-ary relation QUOTIENTMAP. If TOPSP$[X,T]\wedge$TOPSP$[Y,T']\wedge$ SURJ$[p,X,Y]$ then QUOTIENTMAP$[p,X,T,Y,T']\leftrightarrow(\forall U\subseteq Y)(U\in T'\leftrightarrow$ Cnv$(p)|_{Rng}U\in T)$.

QUOTIENTMAP$[p,X,T,Y,T']\leftrightarrow($TOPSP$[X,T]\wedge$TOPSP$[Y,T']\wedge$SURJ$[p,X,Y]\wedge(\forall U)(\subseteq[U,Y]\rightarrow$ $U\in T'\leftrightarrow|_{\mathtt{Rng}}(Cnv(p),U)\in T))$

```
DEFINITION MunkTop.22.2: 5-ary relation SATURATED. If TOPSP[X,T]
\wedge TOPSP[Y,T'] \wedge SURJ[p,X,Y] \wedge C \subseteq X then
SATURATED[C,X,T,Y,T'] \iff
(\forall y \in Y)(
  INTERSECTS[C,Cnv(p) |_{Rng} {y}] \implies Cnv(p) |_{Rng} {y} \subseteq C
).
```

DEFINITION MunkTop.22.2: 5-ary relation SATURATED. If TOPSP$[X,T]\wedge$TOPSP$[Y,T']\wedge$ SURJ$[p,X,Y]\wedge C\subseteq X$ then SATURATED$[C,X,T,Y,T']\leftrightarrow(\forall y\in Y)$ $($INTERSECTS$[C,$ Cnv$(p)|_{Rng}\{y\}]\rightarrow$ Cnv$(p)|_{Rng}\{y\}\subseteq C)$.

$\texttt{SATURATED}[C, X, T, Y, T'] \leftrightarrow (\texttt{TOPSP}[X, T] \wedge \texttt{TOPSP}[Y, T'] \wedge \texttt{SURJ}[p, X, Y] \wedge \subseteq[C, X] \wedge (\forall y)(y \in Y \rightarrow \texttt{INTERSECTS}[C, |_{\texttt{Rng}}(\texttt{Cnv}(p), (\iota x_0)(\forall y_0)(y_0 \in x_0 \leftrightarrow y_0 = y))] \rightarrow \subseteq[|_{\texttt{Rng}}(\texttt{Cnv}(p), (\iota x_0)(\forall y_0)(y_0 \in x_0 \leftrightarrow y_0 = y)), C]))$

```
DEFINITION MunkTop.22.3: 5-ary relation OPENMAP. If TOPSP[X,T] \wedge
TOPSP[Y,T'] \wedge FCN[p,X,Y] then OPENMAP[f,X,T,Y,T'] \iff
(\forall U \in T)(
  f |_{Rng} U \in T'
).
```

DEFINITION MunkTop.22.3: 5-ary relation $\texttt{OPENMAP}$. If $\texttt{TOPSP}[X, T] \wedge \texttt{TOPSP}[Y, T'] \wedge \texttt{FCN}[p, X, Y]$ then $\texttt{OPENMAP}[f, X, T, Y, T'] \leftrightarrow (\forall U \in T)(f|_{Rng} U \in T')$.

$\texttt{OPENMAP}[f, X, T, Y, T'] \leftrightarrow (\texttt{TOPSP}[X, T] \wedge \texttt{TOPSP}[Y, T'] \wedge \texttt{FCN}[p, X, Y] \wedge (\forall U)(U \in T \rightarrow |_{\texttt{Rng}}(f, U) \in T'))$

```
DEFINITION MunkTop.22.4: 5-ary relation CLOSEDMAP. If TOPSP[X,T] \wedge
TOPSP[Y,T'] \wedge FCN[p,X,Y] then CLOSEDMAP[f,X,T,Y,T'] \iff
(\forall A : CLOSEDSET[A,X,T])(
  CLOSEDSET[f |_{Rng} A,Y,T']
).
```

DEFINITION MunkTop.22.4: 5-ary relation $\texttt{CLOSEDMAP}$. If $\texttt{TOPSP}[X, T] \wedge \texttt{TOPSP}[Y, T'] \wedge \texttt{FCN}[p, X, Y]$ then $\texttt{CLOSEDMAP}[f, X, T, Y, T'] \leftrightarrow (\forall A : \texttt{CLOSEDSET}[A, X, T])$ $(\texttt{CLOSEDSET}[f|_{Rng} A, Y, T'])$.

$\texttt{CLOSEDMAP}[f, X, T, Y, T'] \leftrightarrow (\texttt{TOPSP}[X, T] \wedge \texttt{TOPSP}[Y, T'] \wedge \texttt{FCN}[p, X, Y] \wedge (\forall A)$ $(\texttt{CLOSEDSET}[A, X, T] \rightarrow \texttt{CLOSEDSET}[|_{\texttt{Rng}}(f, A), Y, T']))$

```
DEFINITION MunkTop.22.5: 3-ary function Quotienttop. If TOPSP[X,T] \wedge
SURJ[p,X,A] then Quotienttop(p,X,A) \simeq (!T')(
  TOPOLOGY[T',A] \wedge QUOTIENTMAP[p,X,T,A,T']
).
```

DEFINITION MunkTop.22.5: 3-ary function Quotienttop. If TOPSP$[X,T]\wedge$SURJ$[p,X,A]$ then Quotienttop$(p,X,A) \simeq (!T')$(TOPOLOGY$[T',A] \wedge$ QUOTIENTMAP$[p,X,T,A,T']$).

Quotienttop$(p,X,A) \simeq (\iota y_0)$(TOPSP$[X,T] \wedge$ SURJ$[p,X,A] \wedge y_0 \simeq (\iota x_0)(\exists T')(x_0 = T' \wedge$ (TOPOLOGY$[T',A] \wedge$ QUOTIENTMAP$[p,X,T,A,T']$)))

```
DEFINITION MunkTop.22.6: 2-ary relation QUOTIENTSPACE. If TOPSP[X,T]
\wedge PART[X*,X] \wedge SURJ[p,X,X*] \wedge
(\forall x \in X)(p(x) = (!w \in X*)(x \in w)) \wedge
T = Quotienttop(p,X,X*) then
QUOTIENTSPACE[X*,T] \iff \top.
```

DEFINITION MunkTop.22.6: 2-ary relation QUOTIENTSPACE. If TOPSP$[X,T]\wedge$PART$[X^*,X]\wedge$ SURJ$[p,X,X^*] \wedge (\forall x \in X)(p(x) = (!w \in X^*)(x \in w)) \wedge T =$ Quotienttop(p,X,X^*) then QUOTIENTSPACE$[X^*,T] \leftrightarrow \top$.

QUOTIENTSPACE$[X^*,T] \leftrightarrow$ (TOPSP$[X,T] \wedge$ PART$[X^*,X] \wedge$ SURJ$[p,X,X^*] \wedge (\forall x)(x \in X \rightarrow (\iota x_0)(\varpi_0(x,x_0) \in p) = (\iota x_0)(\exists w)(x_0 = w \wedge (x \in w) \wedge (w \in X^*))) \wedge T =$ Quotienttop$(p,X,X^*) \wedge \top$)

G.12 Section 23

```
DEFINITION MunkTop.23.1: 4-ary relation SEPARATION. If TOPSP[X,T]
then SEPARATION[U,V,X,T] \iff U,V \in T \wedge DISJOINT[U,V]
\wedge U,V \neq \varnothing
\wedge U \cup V = X.
```

DEFINITION MunkTop.23.1: 4-ary relation SEPARATION. If TOPSP$[X,T]$ then SEPARATION$[U,V,X,T] \leftrightarrow U,V \in T \wedge$ DISJOINT$[U,V] \wedge U,V \neq \varnothing \wedge U \cup V = X$.

SEPARATION$[U,V,X,T] \leftrightarrow$ (TOPSP$[X,T] \wedge U \in T \wedge V \in T \wedge$ DISJOINT$[U,V] \wedge \neg(U = \varnothing) \wedge \neg(V = \varnothing) \wedge \cup(U,V) = X$)

```
DEFINITION MunkTop.23.2: 2-ary relation CONNECTED. If TOPSP[X,T] then
CONNECTED[X,T] \iff \neg(\exists U,V)(SEPARATION[U,V,X,T]).
```

DEFINITION MunkTop.23.2: 2-ary relation CONNECTED. If TOPSP$[X,T]$ then CONNECTED$[X,T] \leftrightarrow \neg(\exists U,V)$(SEPARATION$[U,V,X,T]$).

CONNECTED$[X,T] \leftrightarrow$ (TOPSP$[X,T] \wedge \neg((\exists U,V)$ SEPARATION$[U,V,X,T]))$

```
DEFINITION MunkTop.23.3: 2-ary relation TOTALLYDISCONNECTED.
If TOPSP[X,T] then TOTALLYDISCONNECTED[X,T] \iff
(\forall U \subseteq X)(
  CONNECTED[U,Subspacetop(U,X,T)] \implies U \approx_{C} 1_{N}
).
```

DEFINITION MunkTop.23.3: 2-ary relation TOTALLYDISCONNECTED. If TOPSP$[X,T]$ then TOTALLYDISCONNECTED$[X,T] \leftrightarrow (\forall U \subseteq X)$(CONNECTED$[U,$ Subspacetop$(U,X,T)] \rightarrow U \approx_{\mathtt{C}} 1_{\mathtt{N}}$).

TOTALLYDISCONNECTED$[X,T] \leftrightarrow$ (TOPSP$[X,T] \wedge (\forall U)(\subseteq[U,X] \rightarrow$ CONNECTED$[U,$ Subspacetop$(U,X,T)] \rightarrow \approx_{\mathtt{C}}[U,1_{\mathtt{N}}]))$

G.13 Section 24

```
DEFINITION MunkTop.24.1: 2-ary relation LINEARCONTINUUM. If SSORD[R,X]
\wedge 1_{N} <_{C} X then LINEARCONTINUUM[R,X] \iff
(\forall U \subseteq X)(
  U \neq \varnothing \wedge (\exists u \in X)(UB[u,R,U]) \implies
  (\exists u \in X)(SUP[u,R,U])
) \wedge
(\forall x,y \in X)(
  x \infixrl{R} y \implies
  (\exists z \in X)(x \infixrl{R} z \infixrl{R} y)
).
```

DEFINITION MunkTop.24.1: 2-ary relation LINEARCONTINUUM. If SSORD$[R, X] \wedge 1_{\mathbb{N}} <_{\mathbb{C}} X$ then LINEARCONTINUUM$[R, X] \leftrightarrow (\forall U \subseteq X)(U \neq \varnothing \wedge (\exists u \in X)(\mathtt{UB}[u, R, U]) \rightarrow (\exists u \in X)(\mathtt{SUP}[u, R, U])) \wedge (\forall x, y \in X)(xRy \rightarrow (\exists z \in X)(xRzRy))$.

LINEARCONTINUUM$[R, X] \leftrightarrow (\mathtt{SSORD}[R, X] \wedge <_{\mathbb{C}}[1_{\mathbb{N}}, X] \wedge (\forall U)(\subseteq [U, X] \rightarrow \neg(U = \varnothing) \wedge (\exists u)(u \in X \wedge \mathtt{UB}[u, R, U]) \rightarrow (\exists u)(u \in X \wedge \mathtt{SUP}[u, R, U])) \wedge (\forall x, y)(x \in X \wedge y \in X \rightarrow \varpi_0(x, y) \in R \rightarrow (\exists z)(z \in X \wedge \varpi_0(x, z) \in R \wedge \varpi_0(z, y) \in R)))$

```
DEFINITION MunkTop.24.2: 5-ary relation PATH. If TOPSP[X,T] \wedge
x,y \in X then PATH[f,X,T,x,y] \iff (\exists a,b \in \mathbb{R})(
  CONTINUOUS[f,
    Ccinterval(a,b,\mathbb{R},{<u,v> : u <_{R} v}),
    Subspacetop(Ccinterval(a,b,\mathbb{R},
                           {<u,v> : u <_{R} v}),
                \mathbb{R},
                Stdrealtop),
    X,T
  ] \wedge
  f(a) = x \wedge f(b) = y
).
```

DEFINITION MunkTop.24.2: 5-ary relation PATH. If TOPSP$[X, T] \wedge x, y \in X$ then PATH$[f, X, T, x, y] \leftrightarrow (\exists a, b \in \mathbb{R})(\mathtt{CONTINUOUS}[f, \mathtt{Ccinterval}(a, b, \mathbb{R}, \{\langle u, v\rangle : u <_{\mathtt{R}} v\}), \mathtt{Subspacetop}(\mathtt{Ccinterval}(a, b, \mathbb{R}, \{\langle u, v\rangle : u <_{\mathtt{R}} v\}), \mathbb{R}, \mathtt{Stdrealtop}), X, T] \wedge f(a) = x \wedge f(b) = y)$.

PATH$[f, X, T, x, y] \leftrightarrow (\mathtt{TOPSP}[X, T] \wedge x \in X \wedge y \in X \wedge (\exists a, b)(a \in \mathbb{R} \wedge b \in \mathbb{R} \wedge \mathtt{CONTINUOUS}[f, \mathtt{Ccinterval}(a, b, \mathbb{R}, (\iota y_0)(\forall x_0)(x_0 \in y_0 \leftrightarrow (\exists u, v)(x_0 = \varpi_0(u, v) \wedge (<_{\mathtt{R}}[u, v])))), \mathtt{Subspacetop}(\mathtt{Ccinterval}(a, b, \mathbb{R}, (\iota y_0)(\forall x_0)(x_0 \in y_0 \leftrightarrow (\exists u, v)(x_0 = \varpi_0(u, v) \wedge (<_{\mathtt{R}}[u, v])))), \mathbb{R}, \mathtt{Stdrealtop}), X, T] \wedge (\iota x_0)(\varpi_0(a, x_0) \in f) = x \wedge (\iota x_0)(\varpi_0(b, x_0) \in f) = y))$

```
DEFINITION MunkTop.24.3: 2-ary relation PATHCONNECTED. If TOPSP[X,T] then
PATHCONNECTED[X,T] \iff (\forall x,y \in X)(\exists f)(PATH[f,X,T,x,y]).
```

DEFINITION MunkTop.24.3: 2-ary relation PATHCONNECTED. If TOPSP$[X,T]$ then PATHCONNECTED$[X,T] \leftrightarrow (\forall x,y \in X)(\exists f)(\texttt{PATH}[f,X,T,x,y])$.

PATHCONNECTED$[X,T] \leftrightarrow (\texttt{TOPSP}[X,T] \wedge (\forall x,y)(x \in X \wedge y \in X \rightarrow (\exists f)\,\texttt{PATH}[f,X,T,x,y]))$

```
DEFINITION MunkTop.24.4: 1-ary function Rnunitball. If n \in \mathbb{N}
then Rnunitball(n) \simeq
{x \in Cartespow(\mathbb{R},n) : Rnnorm(n,x) \leq_{R} 1_{R}}.
```

DEFINITION MunkTop.24.4: 1-ary function Rnunitball. If $n \in \mathbb{N}$ then Rnunitball$(n) \simeq \{x \in \texttt{Cartespow}(\mathbb{R},n) : \texttt{Rnnorm}(n,x) \leq_{\texttt{R}} 1_{\texttt{R}}\}$.

Rnunitball$(n) \simeq (\iota z_0)(n \in \mathbb{N} \wedge z_0 \simeq (\iota y_0)(\forall x_0)(x_0 \in y_0 \leftrightarrow (\exists x)$
$(x_0 = x \wedge (\leq_{\texttt{R}}[\texttt{Rnnorm}(n,x), 1_{\texttt{R}}]) \wedge (x \in \texttt{Cartespow}(\mathbb{R},n)))))$

```
DEFINITION MunkTop.24.5: 1-ary function Puncturedeuclideanspace.
If n \in \mathbb{N} then Puncturedeuclideanspace(n) \simeq
< Cartespow(\mathbb{R},n) \less {0_{R}},
  Subspacetop(
    Cartespow(\mathbb{R},n) \less {0_{R}},
    Cartespow(\mathbb{R},n),
    Cartespow(Stdrealtop,n)
  )
>.
```

DEFINITION MunkTop.24.5: 1-ary function Puncturedeuclideanspace. If $n \in \mathbb{N}$ then Puncturedeuclideanspace$(n) \simeq \langle \texttt{Cartespow}(\mathbb{R},n) \backslash \{0_{\texttt{R}}\},$ Subspacetop$(\texttt{Cartespow}(\mathbb{R},n) \backslash \{0_{\texttt{R}}\}, \texttt{Cartespow}(\mathbb{R},n), \texttt{Cartespow}(\texttt{Stdrealtop},n))\rangle$.

Puncturedeuclideanspace$(n) \simeq (\iota z_0)(n \in \mathbb{N} \wedge z_0 \simeq \varpi_0(\backslash(\texttt{Cartespow}(\mathbb{R},n), (\iota x_0)(\forall y_0)(y_0 \in$ $x_0 \leftrightarrow y_0 = 0_{\texttt{R}})), \texttt{Subspacetop}(\backslash(\texttt{Cartespow}(\mathbb{R},n), (\iota x_0)(\forall y_0)(y_0 \in x_0 \leftrightarrow y_0 = 0_{\texttt{R}})),$ $\texttt{Cartespow}(\mathbb{R},n), \texttt{Cartespow}(\texttt{Stdrealtop},n))))$

```
DEFINITION MunkTop.24.6: 1-ary function Unitsphere. If n \in \mathbb{N}
\wedge n >_{N} 0_{N} then Unitsphere(n) \simeq
{x \in Cartespow(\mathbb{R},n) : Rnnorm(n,x) = 1_{R}}.
```

DEFINITION MunkTop.24.6: 1-ary function `Unitsphere`. If $n \in \mathbb{N} \wedge n >_{\mathtt{N}} 0_{\mathtt{N}}$ then $\mathtt{Unitsphere}(n) \simeq \{x \in \mathtt{Cartespow}(\mathbb{R}, n) : \mathtt{Rnnorm}(n, x) = 1_{\mathtt{R}}\}$.

$\mathtt{Unitsphere}(n) \simeq (\iota z_0)(n \in \mathbb{N} \wedge >_{\mathtt{N}}[n, 0_{\mathtt{N}}] \wedge z_0 \simeq (\iota y_0)(\forall x_0)(x_0 \in y_0 \leftrightarrow (\exists x)(x_0 = x \wedge (\mathtt{Rnnorm}(n, x) = 1_{\mathtt{R}}) \wedge (x \in \mathtt{Cartespow}(\mathbb{R}, n)))))$

G.14 Section 25

```
DEFINITION MunkTop.25.1: 2-ary function Ccequiv. If TOPSP[X,T] then
Ccequiv(X,T) \simeq {<x,y> \in Cartespow(X,2):
  (\exists U : SUBSPACE[U,Subspacetop(U,X,T),X,T])(
    CONNECTED[U,Subspacetop(U,X,T)] \wedge x,y \in U
  )
}.
```

DEFINITION MunkTop.25.1: 2-ary function `Ccequiv`. If $\mathtt{TOPSP}[X, T]$ then $\mathtt{Ccequiv}(X, T) \simeq \{\langle x, y\rangle \in \mathtt{Cartespow}(X, 2) : (\exists U : \mathtt{SUBSPACE}[U, \mathtt{Subspacetop}(U, X, T), X, T])\ (\mathtt{CONNECTED}[U, \mathtt{Subspacetop}(U, X, T)] \wedge x, y \in U)\}$.

$\mathtt{Ccequiv}(X, T) \simeq (\iota z_0)(\mathtt{TOPSP}[X, T] \wedge z_0 \simeq (\iota y_0)(\forall x_0)(x_0 \in y_0 \leftrightarrow (\exists x, y)(x_0 = \varpi_0(x, y) \wedge ((\exists U)(\mathtt{SUBSPACE}[U, \mathtt{Subspacetop}(U, X, T), X, T] \wedge \mathtt{CONNECTED}[U, \mathtt{Subspacetop}(U, X, T)] \wedge x \in U \wedge y \in U)) \wedge (\varpi_0(x, y) \in \mathtt{Cartespow}(X, 2_{\mathtt{N}})))))$

```
DEFINITION MunkTop.25.2: Infix relation \sim_{CC}. If TOPSP[X,T]
\wedge x,y \in X then x \sim_{CC} y \iff <x,y> \in Ccequiv(X,T).
```

DEFINITION MunkTop.25.2: Infix relation $\sim_{\mathtt{CC}}$. If $\mathtt{TOPSP}[X, T] \wedge x, y \in X$ then $x \sim_{\mathtt{CC}} y \leftrightarrow \langle x, y\rangle \in \mathtt{Ccequiv}(X, T)$.

$\sim_{\mathtt{CC}}[x, y] \leftrightarrow (\mathtt{TOPSP}[X, T] \wedge x \in X \wedge y \in X \wedge \varpi_0(x, y) \in \mathtt{Ccequiv}(X, T))$

```
DEFINITION MunkTop.25.3: 2-ary function Components. If TOPSP[X,T] then
Components(X,T) \simeq Part(Ccequiv(X,T)).
```

DEFINITION MunkTop.25.3: 2-ary function `Components`. If $\texttt{TOPSP}[X, T]$ then $\texttt{Components}(X, T) \simeq \texttt{Part}(\texttt{Ccequiv}(X, T))$.

$\texttt{Components}(X, T) \simeq (\iota x_0)(\texttt{TOPSP}[X, T] \wedge x_0 \simeq \texttt{Part}(\texttt{Ccequiv}(X, T)))$

```
DEFINITION MunkTop.25.4: 2-ary function Pcequiv. If TOPSP[X,T] then
Pcequiv(X,T) \simeq {<x,y> \in Cartespow(X,2):
  (\exists f)(
    PATH[f,X,T,x,y]
  )
}.
```

DEFINITION MunkTop.25.4: 2-ary function `Pcequiv`. If $\texttt{TOPSP}[X, T]$ then $\texttt{Pcequiv}(X, T) \simeq \{\langle x, y\rangle \in \texttt{Cartespow}(X, 2) : (\exists f)(\texttt{PATH}[f, X, T, x, y])\}$.

$\texttt{Pcequiv}(X, T) \simeq (\iota z_0)(\texttt{TOPSP}[X, T] \wedge z_0 \simeq (\iota y_0)(\forall x_0)(x_0 \in y_0 \leftrightarrow (\exists x, y)(x_0 = \varpi_0(x, y) \wedge ((\exists f)\, \texttt{PATH}[f, X, T, x, y]) \wedge (\varpi_0(x, y) \in \texttt{Cartespow}(X, 2_{\mathbb{N}})))))$

```
DEFINITION MunkTop.25.5: Infix relation \sim_{PC}. If TOPSP[X,T]
\wedge x,y \in X then x \sim_{PC} y \iff <x,y> \in Pcequiv(X,T).
```

DEFINITION MunkTop.25.5: Infix relation $\sim_{\texttt{PC}}$. If $\texttt{TOPSP}[X, T] \wedge x, y \in X$ then $x \sim_{\texttt{PC}} y \leftrightarrow \langle x, y\rangle \in \texttt{Pcequiv}(X, T)$.

$\sim_{\texttt{PC}}[x, y] \leftrightarrow (\texttt{TOPSP}[X, T] \wedge x \in X \wedge y \in X \wedge \varpi_0(x, y) \in \texttt{Pcequiv}(X, T))$

```
DEFINITION MunkTop.25.6: 2-ary function Pathcomponents. If TOPSP[X,T] then
Pathcomponents(X,T) \simeq Part(Pcequiv(X,T)).
```

DEFINITION MunkTop.25.6: 2-ary function Pathcomponents. If TOPSP$[X,T]$ then Pathcomponents$(X,T) \simeq$ Part(Pcequiv(X,T)).

Pathcomponents$(X,T) \simeq (\iota x_0)($TOPSP$[X,T] \wedge x_0 \simeq$ Part(Pcequiv$(X,T)))$

```
DEFINITION MunkTop.25.7: 3-ary relation LOCALCONNAT. If TOPSP[X,T]
\wedge x \in X then LOCALCONNAT[X,T,x] \iff
(\forall U : NBHD[U,x,X,T])(\exists V \subseteq U)(
  NBHD[V,x,X,T] \wedge CONNECTED[V,Subspacetop(V,X,T)]
).
```

DEFINITION MunkTop.25.7: 3-ary relation LOCALCONNAT. If TOPSP$[X,T] \wedge x \in X$ then LOCALCONNAT$[X,T,x] \leftrightarrow (\forall U$: NBHD$[U,x,X,T])(\exists V \subseteq U)($NBHD$[V,x,X,T] \wedge$ CONNECTED$[V,$ Subspacetop$(V,X,T)])$.

LOCALCONNAT$[X,T,x] \leftrightarrow ($TOPSP$[X,T] \wedge x \in X \wedge (\forall U)($NBHD$[U,x,X,T] \rightarrow (\exists V)(\subseteq [V,U] \wedge$ NBHD$[V,x,X,T] \wedge$ CONNECTED$[V,$ Subspacetop$(V,X,T)])))$

```
DEFINITION MunkTop.25.8: 2-ary relation LOCALCONN. If TOPSP[X,T] then
LOCALCONN[X,T] \iff (\forall x \in X)(LOCALCONNAT[X,T,x]).
```

DEFINITION MunkTop.25.8: 2-ary relation LOCALCONN. If TOPSP$[X,T]$ then LOCALCONN$[X,T] \leftrightarrow (\forall x \in X)($LOCALCONNAT$[X,T,x])$.

LOCALCONN$[X,T] \leftrightarrow ($TOPSP$[X,T] \wedge (\forall x)(x \in X \rightarrow$ LOCALCONNAT$[X,T,x]))$

```
DEFINITION MunkTop.25.9: 3-ary relation LOCALPATHCONNAT. If TOPSP[X,T]
\wedge x \in X then LOCALPATHCONNAT[X,T,x] \iff
(\forall U : NBHD[U,x,X,T])(\exists V \subseteq U)(
  NBHD[V,x,X,T] \wedge PATHCONNECTED[V,Subspacetop(V,X,T)]
).
```

DEFINITION MunkTop.25.9: 3-ary relation LOCALPATHCONNAT. If TOPSP$[X,T] \wedge x \in X$ then LOCALPATHCONNAT$[X,T,x] \leftrightarrow (\forall U :$ NBHD$[U,x,X,T])(\exists V \subseteq U)($NBHD$[V,x,X,T] \wedge$ PATHCONNECTED$[V,$ Subspacetop$(V,X,T)])$.

LOCALPATHCONNAT$[X,T,x] \leftrightarrow ($TOPSP$[X,T] \wedge x \in X \wedge (\forall U)($NBHD$[U,x,X,T] \rightarrow (\exists V)(\subseteq[V,U] \wedge$ NBHD$[V,x,X,T] \wedge$ PATHCONNECTED$[V,$ Subspacetop$(V,X,T)])))$

```
DEFINITION MunkTop.25.10: 2-ary relation LOCALPATHCONN. If TOPSP[X,T] then
LOCALPATHCONN[X,T] \iff (\forall x \in X)(LOCALPATHCONNAT[X,T,x]).
```

DEFINITION MunkTop.25.10: 2-ary relation LOCALPATHCONN. If TOPSP$[X,T]$ then LOCALPATHCONN$[X,T] \leftrightarrow (\forall x \in X)($LOCALPATHCONNAT$[X,T,x])$.

LOCALPATHCONN$[X,T] \leftrightarrow ($TOPSP$[X,T] \wedge (\forall x)(x \in X \rightarrow$ LOCALPATHCONNAT$[X,T,x]))$

```
DEFINITION MunkTop.25.11: 3-ary relation WEAKLOCALCONNAT. If TOPSP[X,T]
\wedge x \in X then WEAKLOCALCONNAT[X,T,x] \iff
(\forall U : NBHD[U,x,X,T])(\exists V \subseteq U)(
  CONNECTED[V,Subspacetop(V,X,T)] \wedge (\exists W \subseteq V)(
    NBHD[W,x,X,T]
  )
).
```

DEFINITION MunkTop.25.11: 3-ary relation WEAKLOCALCONNAT. If TOPSP$[X,T] \wedge x \in X$ then WEAKLOCALCONNAT$[X,T,x] \leftrightarrow (\forall U :$ NBHD$[U,x,X,T])(\exists V \subseteq U)$ (CONNECTED$[V,$ Subspacetop$(V,X,T)] \wedge (\exists W \subseteq V)($NBHD$[W,x,X,T]))$.

WEAKLOCALCONNAT$[X,T,x] \leftrightarrow ($TOPSP$[X,T] \wedge x \in X \wedge (\forall U)($NBHD$[U,x,X,T] \rightarrow (\exists V)$ $(\subseteq[V,U] \wedge$CONNECTED$[V,$ Subspacetop$(V,X,T)] \wedge (\exists W)(\subseteq[W,V] \wedge$NBHD$[W,x,X,T])))$

G.15 Section 26

```
DEFINITION MunkTop.26.1: 2-ary relation COVER. If \mathscr{A}
\subseteq \wp(X) then COVER[\mathscr{A},X] \iff
\cup(\mathscr{A}) = X.
```

DEFINITION MunkTop.26.1: 2-ary relation COVER. If $\mathscr{A} \subseteq \wp(X)$ then $\texttt{COVER}[\mathscr{A}, X] \leftrightarrow \cup(\mathscr{A}) = X$.

$\texttt{COVER}[\mathscr{A}, X] \leftrightarrow (\subseteq[\mathscr{A}, \wp(X)] \wedge \cup(\mathscr{A}) = X)$

```
DEFINITION MunkTop.26.2: 3-ary relation OPENCOVER. If TOPSP[X,T] then
OPENCOVER[A,X,T] \iff COVER[A,X] \wedge A \subseteq T.
```

DEFINITION MunkTop.26.2: 3-ary relation OPENCOVER. If $\texttt{TOPSP}[X, T]$ then $\texttt{OPENCOVER}[A, X, T] \leftrightarrow \texttt{COVER}[A, X] \wedge A \subseteq T$.

$\texttt{OPENCOVER}[A, X, T] \leftrightarrow (\texttt{TOPSP}[X, T] \wedge \texttt{COVER}[A, X] \wedge \subseteq[A, T])$

```
DEFINITION MunkTop.26.3: 2-ary relation COMPACT. If TOPSP[X,T] then
COMPACT[X,T] \iff (\forall A : OPENCOVER[A,X,T])(\exists C \subseteq A)(
  OPENCOVER[C,X,T] \wedge DFIN[C]
).
```

DEFINITION MunkTop.26.3: 2-ary relation COMPACT. If $\texttt{TOPSP}[X, T]$ then $\texttt{COMPACT}[X, T] \leftrightarrow (\forall A : \texttt{OPENCOVER}[A, X, T])(\exists C \subseteq A)(\texttt{OPENCOVER}[C, X, T] \wedge \texttt{DFIN}[C])$.

$\texttt{COMPACT}[X, T] \leftrightarrow (\texttt{TOPSP}[X, T] \wedge (\forall A)(\texttt{OPENCOVER}[A, X, T] \rightarrow (\exists C)(\subseteq[C, A] \wedge \texttt{OPENCOVER}[C, X, T] \wedge \texttt{DFIN}[C])))$

```
DEFINITION MunkTop.26.4: 3-ary relation COVER. If \mathscr{A}
\subseteq \wp(X) \wedge Y \subseteq X then
COVER[Y,\mathscr{A},X] \iff
\cup(\mathscr{A}) \supseteq Y.
```

DEFINITION MunkTop.26.4: 3-ary relation COVER. If $\mathscr{A} \subseteq \wp(X) \wedge Y \subseteq X$ then $\texttt{COVER}[Y, \mathscr{A}, X] \leftrightarrow \cup(\mathscr{A}) \supseteq Y$.

$\texttt{COVER}[Y, \mathscr{A}, X] \leftrightarrow (\subseteq[\mathscr{A}, \wp(X)] \wedge \subseteq[Y, X] \wedge \supseteq[\cup(\mathscr{A}), Y])$

```
DEFINITION MunkTop.26.5: 1-ary relation FINITEINTERSPROP.
FINITEINTERSPROP[C] \iff
(\forall F \subseteq C)(
  DFIN[F] \implies \cap(F) \neq \varnothing
).
```

DEFINITION MunkTop.26.5: 1-ary relation FINITEINTERSPROP.
$\texttt{FINITEINTERSPROP}[C] \leftrightarrow (\forall F \subseteq C)(\texttt{DFIN}[F] \rightarrow \cap(F) \neq \varnothing)$.

$\texttt{FINITEINTERSPROP}[C] \leftrightarrow (\forall F)(\subseteq[F, C] \rightarrow \texttt{DFIN}[F] \rightarrow \neg(\cap(F) = \varnothing))$

G.16 Section 27

```
DEFINITION MunkTop.27.1: 4-ary function Distance. If METRICSPACE[X,T,d]
\wedge A \subseteq X \wedge A \neq \varnothing \wedge x \in X then
Distance(x,A,X,d) \simeq Inf_{R}({d(x,a) : a \in A}).
```

DEFINITION MunkTop.27.1: 4-ary function Distance. If $\texttt{METRICSPACE}[X, T, d] \wedge A \subseteq X \wedge A \neq \varnothing \wedge x \in X$ then $\texttt{Distance}(x, A, X, d) \simeq \texttt{Inf}_{\texttt{R}}(\{d(x, a) : a \in A\})$.

$\texttt{Distance}(x, A, X, d) \simeq (\iota x_1)(\texttt{METRICSPACE}[X, T, d] \wedge \subseteq[A, X] \wedge \neg(A = \varnothing) \wedge x \in X \wedge x_1 \simeq \texttt{Inf}_{\texttt{R}}((\iota z_0)(\forall y_0)(y_0 \in z_0 \leftrightarrow (\exists x, a, d)(y_0 = (\iota x_0)(\varpi_0(\varpi_0(x, a), x_0) \in d) \wedge (a \in A)))))$

```
DEFINITION MunkTop.27.2: 4-ary relation LEBESGUENUM.
If METRICSPACE[X,T,d] \wedge OPENCOVER[A,X,T] \wedge
COMPACT[X,T] then LEBESGUENUM[\delta,A,X,d] \iff
\delta >_{R} 0_{R} \wedge
(\forall Y \subseteq X)(
  Diam(Y) <_{R} \delta \implies (\exists U \in A)(Y \subseteq U)
).
```

DEFINITION MunkTop.27.2: 4-ary relation LEBESGUENUM. If METRICSPACE$[X,T,d] \wedge$ OPENCOVER$[A,X,T] \wedge$ COMPACT$[X,T]$ then LEBESGUENUM$[\delta,A,X,d] \leftrightarrow \delta >_{\mathtt{R}} 0_{\mathtt{R}} \wedge$ $(\forall Y \subseteq X)(\mathtt{Diam}(Y) <_{\mathtt{R}} \delta \rightarrow (\exists U \in A)(Y \subseteq U))$.

LEBESGUENUM$[\delta,A,X,d] \leftrightarrow$ (METRICSPACE$[X,T,d] \wedge$OPENCOVER$[A,X,T] \wedge$COMPACT$[X,T] \wedge$ $>_{\mathtt{R}}[\delta, 0_{\mathtt{R}}] \wedge (\forall Y)(\subseteq[Y,X] \rightarrow <_{\mathtt{R}}[\mathtt{Diam}(Y),\delta] \rightarrow (\exists U)(U \in A \wedge \subseteq[Y,U])))$

```
DEFINITION MunkTop.27.3: 5-ary relation UNIFORMCONT.
If METRICSPACE[X,T,d] \wedge METRICSPACE[Y,T',d'] \wedge
f \in Maps(X,Y) then UNIFORMCONT[f,X,d,Y,d'] \iff
(\forall \veps >_{R} 0_{R})(\exists \delta >_{R} 0_{R})
(\forall x_{0},x_{1} \in X)(
  d(x_{0},x_{1}) <_{R} \veps \implies
  d'(f(x_{0}),f(x_{1})) <_{R} \delta
).
```

DEFINITION MunkTop.27.3: 5-ary relation UNIFORMCONT. If METRICSPACE$[X,T,d] \wedge$ METRICSPACE$[Y,T',d'] \wedge f \in$ Maps(X,Y) then UNIFORMCONT$[f,X,d,Y,d'] \leftrightarrow$ $(\forall \varepsilon >_{\mathtt{R}} 0_{\mathtt{R}})(\exists \delta >_{\mathtt{R}} 0_{\mathtt{R}})(\forall x_0, x_1 \in X)(d(x_0,x_1) <_{\mathtt{R}} \varepsilon \rightarrow d'(f(x_0), f(x_1)) <_{\mathtt{R}} \delta)$.

UNIFORMCONT$[f,X,d,Y,d'] \leftrightarrow$ (METRICSPACE$[X,T,d] \wedge$ METRICSPACE$[Y,T',d'] \wedge f \in$ Maps$(X,Y) \wedge (\forall \varepsilon)(>_{\mathtt{R}}[\varepsilon, 0_{\mathtt{R}}] \rightarrow (\exists \delta)(>_{\mathtt{R}}[\delta, 0_{\mathtt{R}}] \wedge (\forall x_0, x_1)(x_0 \in X \wedge x_1 \in X \rightarrow$ $<_{\mathtt{R}}[(\iota y_0)(\varpi_0(\varpi_0(x_0,x_1),y_0) \in d), \varepsilon] \rightarrow <_{\mathtt{R}}[(\iota z_0)(\varpi_0$ $(\varpi_0((\iota y_0)(\varpi_0(x_0,y_0) \in f), (\iota x_0)(\varpi_0(x_1,x_0) \in f)), z_0) \in d'), \delta])))$

```
DEFINITION MunkTop.27.4: 3-ary relation ISOLATEDPT. If TOPSP[X,T]
\wedge x \in X then ISOLATEDPT[x,X,T] \iff {x} \in T.
```

DEFINITION MunkTop.27.4: 3-ary relation ISOLATEDPT. If TOPSP$[X,T] \wedge x \in X$ then ISOLATEDPT$[x,X,T] \leftrightarrow \{x\} \in T$.

ISOLATEDPT$[x,X,T] \leftrightarrow$ (TOPSP$[X,T] \wedge x \in X \wedge (\iota x_0)(\forall y_0)(y_0 \in x_0 \leftrightarrow y_0 = x) \in T)$

G.17 Section 28

```
DEFINITION MunkTop.28.1: 2-ary relation LIMITPTCPT. If TOPSP[X,T] then
LIMITPTCPT[X,T] \iff (\forall Y \subseteq X)(
  INF[Y] \implies (\exists y \in X)(LIMITPT[y,Y,X,T])
).
```

DEFINITION MunkTop.28.1: 2-ary relation LIMITPTCPT. If TOPSP$[X,T]$ then LIMITPTCPT$[X,T] \leftrightarrow (\forall Y \subseteq X)(\texttt{INF}[Y] \rightarrow (\exists y \in X)(\texttt{LIMITPT}[y,Y,X,T]))$.

LIMITPTCPT$[X,T] \leftrightarrow (\texttt{TOPSP}[X,T] \wedge (\forall Y)(\subseteq[Y,X] \rightarrow \texttt{INF}[Y] \rightarrow (\exists y)(y \in X \wedge \texttt{LIMITPT}[y,Y,X,T])))$

```
DEFINITION MunkTop.28.2: 2-ary relation SUBSEQ. If f \in Maps(\omega,A)
\wedge g \in Maps(\omega,\omega) \wedge h = g \circ f then
SUBSEQ[h,f] \iff (\forall n,m \in \omega)(
  n <_{O} m \implies g(n) <_{O} g(m)
).
```

DEFINITION MunkTop.28.2: 2-ary relation SUBSEQ. If $f \in \texttt{Maps}(\omega, A) \wedge g \in \texttt{Maps}(\omega,\omega) \wedge h = g \circ f$ then $\texttt{SUBSEQ}[h,f] \leftrightarrow (\forall n, m \in \omega)(n <_{\texttt{O}} m \rightarrow g(n) <_{\texttt{O}} g(m))$.

$\texttt{SUBSEQ}[h,f] \leftrightarrow (f \in \texttt{Maps}(\omega, A) \wedge g \in \texttt{Maps}(\omega,\omega) \wedge h = \circ(g,f) \wedge (\forall n, m)(n \in \omega \wedge m \in \omega \rightarrow <_{\texttt{O}}[n,m] \rightarrow <_{\texttt{O}}[(\iota x_0)(\varpi_{\texttt{O}}(n,x_0) \in g), (\iota x_0)(\varpi_{\texttt{O}}(m,x_0) \in g)]))$

```
DEFINITION MunkTop.28.3: 2-ary relation SEQCPT. If TOPSP[X,T] then
SEQCPT[X,T] \iff (\forall f \in Maps(\omega,X))
(\exists g : SUBSEQ[g,f])(
  (\exists x \in X)(TOPCONV[g,x,T])
).
```

DEFINITION MunkTop.28.3: 2-ary relation SEQCPT. If TOPSP$[X,T]$ then SEQCPT$[X,T] \leftrightarrow (\forall f \in \texttt{Maps}(\omega, X))(\exists g : \texttt{SUBSEQ}[g,f])((\exists x \in X)(\texttt{TOPCONV}[g,x,T]))$.

SEQCPT$[X,T] \leftrightarrow (\texttt{TOPSP}[X,T] \wedge (\forall f)(f \in \texttt{Maps}(\omega, X) \rightarrow (\exists g)(\texttt{SUBSEQ}[g,f] \wedge (\exists x)(x \in X \wedge \texttt{TOPCONV}[g,x,T]))))$

```
DEFINITION MunkTop.28.4: 2-ary relation CNTBLCPT. If TOPSP[X,T] then
CNTBLCPT[X,T] \iff (\forall A : OPENCOVER[A,X,T])(
  CNTBL[A] \implies (\exists C \subseteq A)(
    OPENCOVER[C,X,T] \wedge DFIN[C]
  )
).
```

DEFINITION MunkTop.28.4: 2-ary relation CNTBLCPT. If TOPSP$[X,T]$ then CNTBLCPT$[X,T] \leftrightarrow (\forall A :$ OPENCOVER$[A,X,T])($CNTBL$[A] \rightarrow (\exists C \subseteq A)$ (OPENCOVER$[C,X,T] \wedge$ DFIN$[C]))$.

CNTBLCPT$[X,T] \leftrightarrow ($TOPSP$[X,T] \wedge (\forall A)($OPENCOVER$[A,X,T] \rightarrow$ CNTBL$[A] \rightarrow (\exists C)$ $(\subseteq [C,A] \wedge$ OPENCOVER$[C,X,T] \wedge$ DFIN$[C])))$

G.18 Section 29

```
DEFINITION MunkTop.29.1: 3-ary relation LOCCPTAT. If TOPSP[X,T]
\wedge x \in X then LOCCPTAT[X,T,x] \iff (\exists C,V \subseteq X)(
  COMPACT[C,Subspacetop(C,X,T)] \wedge
  NBHD[V,x,X,T] \wedge V \subseteq C
).
```

DEFINITION MunkTop.29.1: 3-ary relation LOCCPTAT. If TOPSP$[X,T] \wedge x \in X$ then LOCCPTAT$[X,T,x] \leftrightarrow (\exists C, V \subseteq X)($COMPACT$[C,$ Subspacetop$(C,X,T)] \wedge$NBHD$[V,x,X,T] \wedge$ $V \subseteq C)$.

LOCCPTAT$[X,T,x] \leftrightarrow ($TOPSP$[X,T] \wedge x \in X \wedge (\exists C,V)(\subseteq [C,X] \wedge \subseteq [V,X] \wedge$ COMPACT$[C,$ Subspacetop$(C,X,T)] \wedge$ NBHD$[V,x,X,T] \wedge \subseteq [V,C]))$

```
DEFINITION MunkTop.29.2: 2-ary relation LOCCPT. If TOPSP[X,T] then
LOCCPT[X,T] \iff (\forall x \in X)(LOCCPTAT[X,T,x]).
```

DEFINITION MunkTop.29.2: 2-ary relation LOCCPT. If TOPSP$[X,T]$ then LOCCPT$[X,T] \leftrightarrow$ $(\forall x \in X)($LOCCPTAT$[X,T,x])$.

LOCCPT$[X,T] \leftrightarrow ($TOPSP$[X,T] \wedge (\forall x)(x \in X \rightarrow$ LOCCPTAT$[X,T,x]))$

```
DEFINITION MunkTop.29.3: 4-ary relation COMPACTIFICATION.
If COMPACT[Y,T'] \wedge HAUSDORFF[Y,T'] \wedge X \neq Y \wedge
SUBSPACE[X,T,Y,T'] \wedge Closure(X,Y,T') = Y then
COMPACTIFICATION[Y,T',X,T] \iff \top.
```

DEFINITION MunkTop.29.3: 4-ary relation COMPACTIFICATION. If COMPACT$[Y, T'] \wedge$ HAUSDORFF$[Y, T'] \wedge X \neq Y \wedge$ SUBSPACE$[X, T, Y, T'] \wedge$ Closure$(X, Y, T') = Y$ then COMPACTIFICATION$[Y, T', X, T] \leftrightarrow \top$.

COMPACTIFICATION$[Y, T', X, T] \leftrightarrow$ (COMPACT$[Y, T'] \wedge$ HAUSDORFF$[Y, T'] \wedge \neg(X = Y) \wedge$ SUBSPACE$[X, T, Y, T'] \wedge$ Closure$(X, Y, T') = Y \wedge \top$)

```
DEFINITION MunkTop.29.4: 2-ary function Oneptcompactification.
If TOPSP[X,T] then Oneptcompactification(X,T) \simeq (!<Y,T'>)(
  COMPACTIFICATION[Y,T',X,T] \wedge Y \less X \approx_{C} 1_{N}
).
```

DEFINITION MunkTop.29.4: 2-ary function Oneptcompactification. If TOPSP$[X, T]$ then Oneptcompactification$(X, T) \simeq (!\langle Y, T'\rangle)$(COMPACTIFICATION$[Y, T', X, T] \wedge Y \backslash X \approx_{\mathtt{C}} 1_{\mathtt{N}}$).

Oneptcompactification$(X, T) \simeq (\iota y_0)$(TOPSP$[X, T] \wedge y_0 \simeq (\iota x_0)(\exists Y, T')(x_0 = \varpi_0(Y, T') \wedge$ (COMPACTIFICATION$[Y, T', X, T] \wedge \approx_{\mathtt{C}}[\backslash(Y, X), 1_{\mathtt{N}}])))$

G.19 Section 30

```
DEFINITION MunkTop.30.1: 2-ary relation SECONDCNTBL. If TOPSP[X,T] then
SECONDCNTBL[X,T] \iff (\exists \mathscr{B})(
  CNTBL[\mathscr{B}] \wedge Basisgentop(\mathscr{B},X) = T
).
```

DEFINITION MunkTop.30.1: 2-ary relation SECONDCNTBL. If TOPSP$[X, T]$ then SECONDCNTBL$[X, T] \leftrightarrow (\exists \mathscr{B})$(CNTBL$[\mathscr{B}] \wedge$ Basisgentop$(\mathscr{B}, X) = T$).

SECONDCNTBL$[X, T] \leftrightarrow$ (TOPSP$[X, T] \wedge (\exists \mathscr{B})$ CNTBL$[\mathscr{B}] \wedge$ Basisgentop$(\mathscr{B}, X) = T$)

```
DEFINITION MunkTop.30.2: 3-ary relation DENSE. If TOPSP[X,T] \wedge
A \subseteq X then DENSE[A,X,T] \iff Closure(A,X,T) = X.
```

DEFINITION MunkTop.30.2: 3-ary relation DENSE. If $\texttt{TOPSP}[X, T] \wedge A \subseteq X$ then $\texttt{DENSE}[A, X, T] \leftrightarrow \texttt{Closure}(A, X, T) = X$.

$\texttt{DENSE}[A, X, T] \leftrightarrow (\texttt{TOPSP}[X, T] \wedge \subseteq[A, X] \wedge \texttt{Closure}(A, X, T) = X)$

```
DEFINITION MunkTop.30.3: 2-ary relation LINDELOF. If TOPSP[X,T] then
LINDELOF[X,T] \iff (\forall A : OPENCOVER[A,X,T])(
  (\exists C \subseteq A)(
    CNTBL[C] \wedge OPENCOVER[C,X,T]
  )
).
```

DEFINITION MunkTop.30.3: 2-ary relation LINDELOF. If $\texttt{TOPSP}[X, T]$ then $\texttt{LINDELOF}[X, T] \leftrightarrow (\forall A : \texttt{OPENCOVER}[A, X, T])((\exists C \subseteq A)(\texttt{CNTBL}[C] \wedge \texttt{OPENCOVER}[C, X, T]))$.

$\texttt{LINDELOF}[X, T] \leftrightarrow (\texttt{TOPSP}[X, T] \wedge (\forall A)(\texttt{OPENCOVER}[A, X, T] \rightarrow (\exists C)(\subseteq[C, A] \wedge \texttt{CNTBL}[C] \wedge \texttt{OPENCOVER}[C, X, T])))$

```
DEFINITION MunkTop.30.4: 2-ary relation SEPARABLESPACE. If TOPSP[X,T]
then SEPARABLESPACE[X,T] \iff (\exists Y \subseteq X)(
  CNTBL[Y] \wedge DENSE[Y,X,T]
).
```

DEFINITION MunkTop.30.4: 2-ary relation SEPARABLESPACE. If $\texttt{TOPSP}[X, T]$ then $\texttt{SEPARABLESPACE}[X, T] \leftrightarrow (\exists Y \subseteq X)(\texttt{CNTBL}[Y] \wedge \texttt{DENSE}[Y, X, T])$.

$\texttt{SEPARABLESPACE}[X, T] \leftrightarrow (\texttt{TOPSP}[X, T] \wedge (\exists Y)(\subseteq[Y, X] \wedge \texttt{CNTBL}[Y] \wedge \texttt{DENSE}[Y, X, T]))$

```
DEFINITION MunkTop.30.5: 0-ary function Sorgenfreyplane.
Sorgenfreyplane \simeq
< \mathbb{R} \times \mathbb{R},
  Lowerlimitrealtop \times Lowerlimitrealtop
>.
```

DEFINITION MunkTop.30.5: 0-ary function Sorgenfreyplane. Sorgenfreyplane $\simeq$ $\langle \mathbb{R} \times \mathbb{R}, \texttt{Lowerlimitrealtop} \times \texttt{Lowerlimitrealtop} \rangle$.

$\texttt{Sorgenfreyplane} \simeq \varpi_0(\times(\mathbb{R}, \mathbb{R}), \times(\texttt{Lowerlimitrealtop}, \texttt{Lowerlimitrealtop}))$

```
DEFINITION MunkTop.30.6: 3-ary relation GDELTA. If TOPSP[X,T] \wedge
A \subseteq X then GDELTA[A,X,T] \iff (\exists S,f)(
  CNTBL[S] \wedge f \in Maps(S,T) \wedge A = \cap({f(s) : s \in S})
).
```

DEFINITION MunkTop.30.6: 3-ary relation GDELTA. If $\texttt{TOPSP}[X, T] \wedge A \subseteq X$ then $\texttt{GDELTA}[A, X, T] \leftrightarrow (\exists S, f)(\texttt{CNTBL}[S] \wedge f \in \texttt{Maps}(S, T) \wedge A = \cap(\{f(s) : s \in S\}))$.

$\texttt{GDELTA}[A, X, T] \leftrightarrow (\texttt{TOPSP}[X, T] \wedge \subseteq[A, X] \wedge (\exists S, f)\, \texttt{CNTBL}[S] \wedge f \in \texttt{Maps}(S, T) \wedge A = \cap((\iota z_0)(\forall y_0)(y_0 \in z_0 \leftrightarrow (\exists s, f)(y_0 = (\iota x_0)(\varpi_0(s, x_0) \in f) \wedge (s \in S)))))$

G.20 Section 31

```
DEFINITION MunkTop.31.1: 2-ary relation REGULARSP. If TOPSP[X,T] then
REGULARSP[X,T] \iff
(\forall x \in X)(CLOSEDSET[{x},X,T]) \wedge
(\forall x \in X)(\forall B \subseteq X)(
  CLOSEDSET[B,X,T] \wedge \neg(x \in B) \implies
  (\exists U,V \in T)(
    DISJOINT[U,V] \wedge x \in U \wedge B \subseteq V
  )
).
```

DEFINITION MunkTop.31.1: 2-ary relation REGULARSP. If $\texttt{TOPSP}[X, T]$ then $\texttt{REGULARSP}[X, T] \leftrightarrow (\forall x \in X)(\texttt{CLOSEDSET}[\{x\}, X, T]) \wedge (\forall x \in X)(\forall B \subseteq X)(\texttt{CLOSEDSET}[B, X, T] \wedge \neg(x \in B) \rightarrow (\exists U, V \in T)(\texttt{DISJOINT}[U, V] \wedge x \in U \wedge B \subseteq V))$.

$\texttt{REGULARSP}[X, T] \leftrightarrow (\texttt{TOPSP}[X, T] \wedge (\forall x)(x \in X \rightarrow \texttt{CLOSEDSET}[(\iota x_0)(\forall y_0)(y_0 \in x_0 \leftrightarrow y_0 = x), X, T]) \wedge (\forall x)(x \in X \rightarrow (\forall B)(\subseteq[B, X] \rightarrow \texttt{CLOSEDSET}[B, X, T] \wedge \neg(x \in B) \rightarrow (\exists U, V)(U \in T \wedge V \in T \wedge \texttt{DISJOINT}[U, V] \wedge x \in U \wedge \subseteq[B, V]))))$

```
DEFINITION MunkTop.31.2: 2-ary relation NORMALSP. If TOPSP[X,T] then
NORMALSP[X,T] \iff
(\forall x \in X)(CLOSEDSET[{x},X,T]) \wedge
(\forall A,B \subseteq X)(
  CLOSEDSET[A,X,T] \wedge CLOSEDSET[B,X,T] \wedge DISJOINT[A,B] \implies
  (\exists U,V \in T)(
    DISJOINT[U,V] \wedge A \subseteq U \wedge B \subseteq V
  )
).
```

DEFINITION MunkTop.31.2: 2-ary relation NORMALSP. If TOPSP$[X, T]$ then NORMALSP$[X, T] \leftrightarrow (\forall x \in X)$(CLOSEDSET$[\{x\}, X, T]) \wedge (\forall A, B \subseteq X)$(CLOSEDSET$[A, X, T] \wedge$CLOSEDSET$[B, X, T] \wedge$ DISJOINT$[A, B] \rightarrow (\exists U, V \in T)$(DISJOINT$[U, V] \wedge A \subseteq U \wedge B \subseteq V)$).

NORMALSP$[X, T] \leftrightarrow$ (TOPSP$[X, T] \wedge (\forall x)(x \in X \rightarrow$ CLOSEDSET$[(\iota x_0)(\forall y_0)(y_0 \in x_0 \leftrightarrow y_0 = x), X, T]) \wedge (\forall A, B)(\subseteq[A, X] \wedge \subseteq[B, X] \rightarrow$ CLOSEDSET$[A, X, T] \wedge$CLOSEDSET$[B, X, T] \wedge$ DISJOINT$[A, B] \rightarrow (\exists U, V)(U \in T \wedge V \in T \wedge$ DISJOINT$[U, V] \wedge \subseteq[A, U] \wedge \subseteq[B, V]))))$

G.21 Section 32

```
DEFINITION MunkTop.32.1: 2-ary relation COMPLETELYNORMALSP.
If TOPSP[X,T] then COMPLETELYNORMALSP[X,T] \iff
(\forall Y \subseteq X)(
  NORMALSP[Y,Subspacetop(Y,X,T)]
).
```

DEFINITION MunkTop.32.1: 2-ary relation COMPLETELYNORMALSP. If TOPSP$[X, T]$ then COMPLETELYNORMALSP$[X, T] \leftrightarrow (\forall Y \subseteq X)$(NORMALSP$[Y,$ Subspacetop$(Y, X, T)]$).

COMPLETELYNORMALSP$[X, T] \leftrightarrow$ (TOPSP$[X, T] \wedge (\forall Y)(\subseteq[Y, X] \rightarrow$ NORMALSP$[Y,$ Subspacetop$(Y, X, T)]))$

G.22 Section 33

```
DEFINITION MunkTop.33.1: 4-ary relation SEPBYCONTFUNC. If TOPSP[X,T] \wedge
A,B \subseteq X then SEPBYCONTFUNC[A,B,X,T] \iff
(\exists f \in Maps(X,Ccinterval(0_{R},1_{R},\mathbb{R},
                                 {<u,v> : u <_{R} v}))(
  CONTINUOUS[
    f,X,T,
    Ccinterval(0_{R},1_{R},\mathbb{R},{<u,v> : u <_{R} v}),
    Subspacetop(
      Ccinterval(0_{R},1_{R},\mathbb{R},{<u,v> : u <_{R} v}),
      \mathbb{R},
      Stdrealtop
    )
  ] \wedge
  f |_{Rng} A = {0_{R}} \wedge
  f |_{Rng} B = {1_{R}}
).
```

DEFINITION MunkTop.33.1: 4-ary relation SEPBYCONTFUNC. If $\texttt{TOPSP}[X, T] \wedge A, B \subseteq X$ then $\texttt{SEPBYCONTFUNC}[A, B, X, T] \leftrightarrow (\exists f \in \texttt{Maps}(X, \texttt{Ccinterval}(\texttt{0}_{\texttt{R}}, \texttt{1}_{\texttt{R}}, \mathbb{R}, \{\langle u, v\rangle : u <_{\texttt{R}} v\})))$ $(\texttt{CONTINUOUS}[f, X, T, \texttt{Ccinterval}(\texttt{0}_{\texttt{R}}, \texttt{1}_{\texttt{R}}, \mathbb{R}, \{\langle u, v\rangle : u <_{\texttt{R}} v\}), \texttt{Subspacetop}($ $\texttt{Ccinterval}(\texttt{0}_{\texttt{R}}, \texttt{1}_{\texttt{R}}, \mathbb{R}, \{\langle u, v\rangle : u <_{\texttt{R}} v\}), \mathbb{R}, \texttt{Stdrealtop})] \wedge f|_{Rng}A = \{\texttt{0}_{\texttt{R}}\} \wedge f|_{Rng}B = \{\texttt{1}_{\texttt{R}}\})$.

$\texttt{SEPBYCONTFUNC}[A, B, X, T] \leftrightarrow (\texttt{TOPSP}[X, T] \wedge \subseteq[A, X] \wedge \subseteq[B, X] \wedge (\exists f)(f \in$ $\texttt{Maps}(X, \texttt{Ccinterval}(\texttt{0}_{\texttt{R}}, \texttt{1}_{\texttt{R}}, \mathbb{R}, (\iota y_0)(\forall x_0)(x_0 \in y_0 \leftrightarrow (\exists u, v)(x_0 = \varpi_{\texttt{0}}(u, v) \wedge (<_{\texttt{R}}[u, v])))))) \wedge$ $\texttt{CONTINUOUS}[f, X, T, \texttt{Ccinterval}(\texttt{0}_{\texttt{R}}, \texttt{1}_{\texttt{R}}, \mathbb{R}, (\iota y_0)(\forall x_0)(x_0 \in y_0 \leftrightarrow (\exists u, v)(x_0 = \varpi_{\texttt{0}}(u, v) \wedge$ $(<_{\texttt{R}}[u, v])))), \texttt{Subspacetop}(\texttt{Ccinterval}(\texttt{0}_{\texttt{R}}, \texttt{1}_{\texttt{R}}, \mathbb{R}, (\iota y_0)(\forall x_0)(x_0 \in y_0 \leftrightarrow (\exists u, v)(x_0 =$ $\varpi_{\texttt{0}}(u, v) \wedge (<_{\texttt{R}}[u, v])))), \mathbb{R}, \texttt{Stdrealtop})] \wedge |_{\texttt{Rng}}(f, A) = (\iota x_0)(\forall y_0)(y_0 \in x_0 \leftrightarrow y_0 =$ $\texttt{0}_{\texttt{R}}) \wedge |_{\texttt{Rng}}(f, B) = (\iota x_0)(\forall y_0)(y_0 \in x_0 \leftrightarrow y_0 = \texttt{1}_{\texttt{R}})))$

```
DEFINITION MunkTop.33.2: 2-ary relation COMPLETELYREGULARSP.
If TOPSP[X,T] then
COMPLETELYREGULARSP[X,T] \iff
(\forall x \in X)(CLOSEDSET[{x},X,T]) \wedge
(\forall x \in X)(\forall A \subseteq X)(
```

```
  \neg(x \in A) \wedge CLOSEDSET[A,X,T] \implies
  SEPBYCONTFUNC[A,{x},X,T]
).
```

DEFINITION MunkTop.33.2: 2-ary relation COMPLETELYREGULARSP. If TOPSP$[X,T]$ then COMPLETELYREGULARSP$[X,T] \leftrightarrow (\forall x \in X)$(CLOSEDSET$[\{x\},X,T]) \wedge (\forall x \in X)$ $(\forall A \subseteq X)(\neg(x \in A) \wedge$ CLOSEDSET$[A,X,T] \rightarrow$ SEPBYCONTFUNC$[A,\{x\},X,T])$.

COMPLETELYREGULARSP$[X,T] \leftrightarrow$ (TOPSP$[X,T] \wedge (\forall x)(x \in X \rightarrow$ CLOSEDSET$[(\iota x_0)(\forall y_0)(y_0 \in x_0 \leftrightarrow y_0 = x),X,T]) \wedge (\forall x)(x \in X \rightarrow (\forall A)(\subseteq[A,X] \rightarrow \neg(x \in A) \wedge$ CLOSEDSET$[A,X,T] \rightarrow$ SEPBYCONTFUNC$[A,(\iota x_0)(\forall y_0)(y_0 \in x_0 \leftrightarrow y_0 = x),X,T])))$

```
DEFINITION MunkTop.33.3: 2-ary relation PERFECTLYNORMALSP. If TOPSP[X,T]
then PERFECTLYNORMALSP[X,T] \iff NORMALSP[X,T] \wedge
(\forall F : CLOSEDSET[F,X,T])(GDELTA[F,X,T]).
```

DEFINITION MunkTop.33.3: 2-ary relation PERFECTLYNORMALSP. If TOPSP$[X,T]$ then PERFECTLYNORMALSP$[X,T] \leftrightarrow$ NORMALSP$[X,T] \wedge (\forall F :$ CLOSEDSET$[F,X,T])$ (GDELTA$[F,X,T]$).

PERFECTLYNORMALSP$[X,T] \leftrightarrow$ (TOPSP$[X,T] \wedge$NORMALSP$[X,T] \wedge (\forall F)$(CLOSEDSET$[F,X,T] \rightarrow$ GDELTA$[F,X,T]))$

G.23 Section 34

```
DEFINITION MunkTop.34.1: 2-ary relation LOCALLYMETRIZABLE. If TOPSP[X,T]
then LOCALLYMETRIZABLE[X,T] \iff (\forall x \in X)(
  (\exists U : NBHD[U,x,X,T]) (
    METRIZABLE[U,Subspacetop(U,X,T)]
  )
).
```

DEFINITION MunkTop.34.1: 2-ary relation LOCALLYMETRIZABLE. If TOPSP$[X,T]$ then LOCALLYMETRIZABLE$[X,T] \leftrightarrow (\forall x \in X)((\exists U :$ NBHD$[U,x,X,T])$ (METRIZABLE$[U,$ Subspacetop$(U,X,T)])))$.

LOCALLYMETRIZABLE$[X,T] \leftrightarrow$ (TOPSP$[X,T] \wedge (\forall x)(x \in X \rightarrow (\exists U)$(NBHD$[U,x,X,T] \wedge$ METRIZABLE$[U,$ Subspacetop$(U,X,T)]))))$

G.24 Section 35

```
DEFINITION MunkTop.35.0: Infix relation EXTENDS. If f \in Maps(A,B) \wedge
f' \in Maps (A',B) \wedge A \subseteq A' then f' EXTENDS f \iff
(\forall a \in A)(f'(a) = f(a)).
```

DEFINITION MunkTop.35.0: Infix relation EXTENDS. If $f \in$ Maps$(A,B) \wedge f' \in$ Maps$(A',B) \wedge A \subseteq A'$ then f' EXTENDS $f \leftrightarrow (\forall a \in A)(f'(a) = f(a))$.

EXTENDS$[f',f] \leftrightarrow (f \in$ Maps$(A,B) \wedge f' \in$ Maps$(A',B) \wedge \subseteq[A,A'] \wedge (\forall a)(a \in A \rightarrow (\iota x_0)(\varpi_0(a,x_0) \in f') = (\iota x_0)(\varpi_0(a,x_0) \in f)))$

```
DEFINITION MunkTop.35.1: 2-ary relation UNIVEXTPROP. If TOPSP[Y,T] then
UNIVEXTPROP[Y,T] \iff (\forall X,T',A,f)(
  NORMALSP[X,T'] \wedge
  CLOSEDSET[A,X,T'] \wedge
  CONTINUOUS[f,A,Subspacetop(A,X,T'),Y,T] \implies
  (\exists f')(
    CONTINUOUS[f',X,T',Y,T] \wedge
    f' EXTENDS f
  )
).
```

DEFINITION MunkTop.35.1: 2-ary relation UNIVEXTPROP. If TOPSP$[Y,T]$ then UNIVEXTPROP$[Y,T] \leftrightarrow (\forall X,T',A,f)$(NORMALSP$[X,T'] \wedge$ CLOSEDSET$[A,X,T'] \wedge$ CONTINUOUS$[f,A,$ Subspacetop$(A,X,T'),Y,T] \rightarrow (\exists f')$(CONTINUOUS$[f',X,T',Y,T] \wedge f'$ EXTENDS $f))$.

UNIVEXTPROP$[Y,T] \leftrightarrow$ (TOPSP$[Y,T] \wedge (\forall X,T',A,f)$ NORMALSP$[X,T'] \wedge$ CLOSEDSET$[A,X,T'] \wedge$ CONTINUOUS$[f,A,$ Subspacetop$(A,X,T'),Y,T] \rightarrow (\exists f')$ CONTINUOUS$[f',X,T',Y,T] \wedge$ EXTENDS$[f',f])$

G.25 Section 36

```
DEFINITION MunkTop.36.1: 1-ary function Manifold. If m \in \mathbb{N}
then Manifold(m) \simeq {<X,T> :
  HAUSDORFF[X,T] \wedge SECONDCNTBL[X,T] \wedge
  (\forall x \in X)(\exists U : NBHD[U,x,X,T])
  (\exists V \in Cartespow(Stdrealtop,m))(
    HOMEOMORPHIC[
      U,Subspacetop(U,X,T),
      V,Subspacetop(V,Cartespow(\mathbb{R},m),Cartespow(Stdrealtop,m))
    ]
  )
}.
```

DEFINITION MunkTop.36.1: 1-ary function $\texttt{Manifold}$. If $m \in \mathbb{N}$ then $\texttt{Manifold}(m) \simeq \{\langle X, T\rangle : \texttt{HAUSDORFF}[X, T] \wedge \texttt{SECONDCNTBL}[X, T] \wedge (\forall x \in X)(\exists U : \texttt{NBHD}[U, x, X, T])(\exists V \in \texttt{Cartespow}(\texttt{Stdrealtop}, m))(\texttt{HOMEOMORPHIC}[U, \texttt{Subspacetop}(U, X, T), V, \texttt{Subspacetop}(V, \texttt{Cartespow}(\mathbb{R}, m), \texttt{Cartespow}(\texttt{Stdrealtop}, m))])\}$.

$\texttt{Manifold}(m) \simeq (\iota z_0)(m \in \mathbb{N} \wedge z_0 \simeq (\iota y_0)(\forall x_0)(x_0 \in y_0 \leftrightarrow (\exists X, T)(x_0 = \varpi_0(X, T) \wedge (\texttt{HAUSDORFF}[X, T] \wedge \texttt{SECONDCNTBL}[X, T] \wedge (\forall x)(x \in X \rightarrow (\exists U)(\texttt{NBHD}[U, x, X, T] \wedge (\exists V)(V \in \texttt{Cartespow}(\texttt{Stdrealtop}, m) \wedge \texttt{HOMEOMORPHIC}[U, \texttt{Subspacetop}(U, X, T), V, \texttt{Subspacetop}(V, \texttt{Cartespow}(\mathbb{R}, m), \texttt{Cartespow}(\texttt{Stdrealtop}, m))])))))))$

```
DEFINITION MunkTop.36.2: 2-ary relation TOPCURVE. TOPCURVE[X,T] \iff
<X,T> \in Manifold(1).
```

DEFINITION MunkTop.36.2: 2-ary relation $\texttt{TOPCURVE}$. $\texttt{TOPCURVE}[X, T] \leftrightarrow \langle X, T\rangle \in \texttt{Manifold}(1)$.

$\texttt{TOPCURVE}[X, T] \leftrightarrow \varpi_0(X, T) \in \texttt{Manifold}(1_{\mathbb{N}})$

```
DEFINITION MunkTop.36.3: 2-ary relation TOPSURFACE. TOPSURFACE[X,T] \iff
<X,T> \in Manifold(2).
```

DEFINITION MunkTop.36.3: 2-ary relation TOPSURFACE. $\texttt{TOPSURFACE}[X,T] \leftrightarrow \langle X,T\rangle \in \texttt{Manifold}(2)$.

$\texttt{TOPSURFACE}[X,T] \leftrightarrow \varpi_0(X,T) \in \texttt{Manifold}(2_{\texttt{N}})$

```
DEFINITION MunkTop.36.4: 1-ary function Support. If TOPSP[X,T] \wedge
\vphi \in Maps(X,\mathbb{R}) then Support(\vphi) \simeq
Closure(
  Cnv(\vphi) |_{Rng} (\mathbb{R} \less {0_{R}}),
  X,T
).
```

DEFINITION MunkTop.36.4: 1-ary function Support. If $\texttt{TOPSP}[X,T] \wedge \varphi \in \texttt{Maps}(X,\mathbb{R})$ then $\texttt{Support}(\varphi) \simeq \texttt{Closure}(\texttt{Cnv}(\varphi)|_{Rng}(\mathbb{R} \setminus \{0_{\texttt{R}}\}), X, T)$.

$\texttt{Support}(\varphi) \simeq (\iota z_0)(\texttt{TOPSP}[X,T] \wedge \varphi \in \texttt{Maps}(X,\mathbb{R}) \wedge z_0 \simeq \texttt{Closure}(|_{\texttt{Rng}}(\texttt{Cnv}(\varphi), (\setminus(\mathbb{R}, (\iota x_0)(\forall y_0)(y_0 \in x_0 \leftrightarrow y_0 = 0_{\texttt{R}})))), X, T))$

```
DEFINITION MunkTop.36.5: 4-ary relation PARTITIONOFUNITY. If TOPSP[X,T]
\wedge I = Ccinterval(0_{R},1_{R},\mathbb{R},{<u,v> : u <_{R} v}) \wedge
T' = Subspacetop(I,\mathbb{R},Stdrealtop) \wedge
n \in \mathbb{N} \wedge \vphi \in Maps(n,Maps(X,I)) \wedge
(\forall i \in n)(CONTINUOUS[\vphi(i),X,T,I,T']) then
PARTITIONOFUNITY[\vphi,n,X,T] \iff
(\forall x \in X)(
  Finitesum_{R}(
    (\lambda i \in n)(\vphi(i)(x))
  ) = 1_{R}
).
```

DEFINITION MunkTop.36.5: 4-ary relation PARTITIONOFUNITY. If $\texttt{TOPSP}[X,T] \wedge I = \texttt{Ccinterval}(0_{\texttt{R}}, 1_{\texttt{R}}, \mathbb{R}, \{\langle u,v\rangle : u <_{\texttt{R}} v\}) \wedge T' = \texttt{Subspacetop}(I, \mathbb{R}, \texttt{Stdrealtop}) \wedge n \in \mathbb{N} \wedge \varphi \in \texttt{Maps}(n, \texttt{Maps}(X,I)) \wedge (\forall i \in n)(\texttt{CONTINUOUS}[\varphi(i), X, T, I, T'])$ then $\texttt{PARTITIONOFUNITY}[\varphi, n, X, T] \leftrightarrow (\forall x \in X)(\texttt{Finitesum}_{\texttt{R}}((\lambda i \in n)(\varphi(i)(x))) = 1_{\texttt{R}})$.

$\texttt{PARTITIONOFUNITY}[\varphi, n, X, T] \leftrightarrow (\texttt{TOPSP}[X, T] \wedge I = \texttt{Ccinterval}(\texttt{0}_{\texttt{R}}, \texttt{1}_{\texttt{R}}, \mathbb{R}, (\iota y_0)(\forall x_0)(x_0 \in y_0 \leftrightarrow (\exists u, v)(x_0 = \varpi_{\texttt{0}}(u, v) \wedge (<_{\texttt{R}}[u, v])))) \wedge T' = \texttt{Subspacetop}(I, \mathbb{R}, \texttt{Stdrealtop}) \wedge n \in \mathbb{N} \wedge \varphi \in \texttt{Maps}(n, \texttt{Maps}(X, I)) \wedge (\forall i)(i \in n \rightarrow \texttt{CONTINUOUS}[(\iota x_0)(\varpi_{\texttt{0}}(i, x_0) \in \varphi), X, T, I, T']) \wedge (\forall x)(x \in X \rightarrow \texttt{Finitesum}_{\texttt{R}}((\iota y_1)(\forall x_1)(x_1 \in y_1 \leftrightarrow (\exists i, z_0)(x_1 = \varpi_{\texttt{0}}(i, z_0) \wedge z_0 = ((\iota y_0)(\varpi_{\texttt{0}}(x, y_0) \in (\iota x_0)(\varpi_{\texttt{0}}(i, x_0) \in \varphi))) \wedge i \in n))) = \texttt{1}_{\texttt{R}}))$

```
DEFINITION MunkTop.36.6: 5-ary relation POUDOMBY.
If PARTITIONOFUNITY[\vphi,n,X,T] \wedge U \in Maps(n,T) \wedge
OPENCOVER[{U(i): i \in n},X,T] then
POUDOMBY[\vphi,n,X,T,U] \iff (\forall i \in n)(
  Support(\vphi(i)) \subseteq U(i)
).
```

DEFINITION MunkTop.36.6: 5-ary relation POUDOMBY. If $\texttt{PARTITIONOFUNITY}[\varphi, n, X, T] \wedge U \in \texttt{Maps}(n, T) \wedge \texttt{OPENCOVER}[\{U(i) : i \in n\}, X, T]$ then $\texttt{POUDOMBY}[\varphi, n, X, T, U] \leftrightarrow (\forall i \in n)(\texttt{Support}(\varphi(i)) \subseteq U(i))$.

$\texttt{POUDOMBY}[\varphi, n, X, T, U] \leftrightarrow (\texttt{PARTITIONOFUNITY}[\varphi, n, X, T] \wedge U \in \texttt{Maps}(n, T) \wedge \texttt{OPENCOVER}[(\iota z_0)(\forall y_0)(y_0 \in z_0 \leftrightarrow (\exists i, U)(y_0 = (\iota x_0)(\varpi_{\texttt{0}}(i, x_0) \in U) \wedge (i \in n))), X, T] \wedge (\forall i)(i \in n \rightarrow \subseteq[\texttt{Support}((\iota x_0)(\varpi_{\texttt{0}}(i, x_0) \in \varphi)), (\iota x_0)(\varpi_{\texttt{0}}(i, x_0) \in U)]))$

```
DEFINITION MunkTop.36.7: 3-ary relation POINTFINFAM.
If A \in Maps(I,\wp(X))
then POINTFINFAM[I,A,X] \iff (\forall x \in X)(
  DFIN[{\alpha \in I : x \in A(\alpha)}]
).
```

DEFINITION MunkTop.36.7: 3-ary relation POINTFINFAM. If $A \in \texttt{Maps}(I, \wp(X))$ then $\texttt{POINTFINFAM}[I, A, X] \leftrightarrow (\forall x \in X)(\texttt{DFIN}[\{\alpha \in I : x \in A(\alpha)\}])$.

$\texttt{POINTFINFAM}[I, A, X] \leftrightarrow (A \in \texttt{Maps}(I, \wp(X)) \wedge (\forall x)(x \in X \rightarrow \texttt{DFIN}[(\iota z_0)(\forall y_0)(y_0 \in z_0 \leftrightarrow (\exists \alpha)(y_0 = \alpha \wedge (x \in (\iota x_0)(\varpi_{\texttt{0}}(\alpha, x_0) \in A)) \wedge (\alpha \in I)))]))$

G.26 Section 37

```
DEFINITION MunkTop.37.1: 1-ary relation CNTBLINTERSPROP.
CNTBLINTERSPROP[\mathscr{A}] \iff
(\forall \mathscr{B} \subseteq \mathscr{A})(
  CNTBL[\mathscr{B}] \implies \cap(\mathscr{B}) \neq \varnothing
).
```

DEFINITION MunkTop.37.1: 1-ary relation CNTBLINTERSPROP.
$\texttt{CNTBLINTERSPROP}[\mathscr{A}] \leftrightarrow (\forall \mathscr{B} \subseteq \mathscr{A})(\texttt{CNTBL}[\mathscr{B}] \rightarrow \cap(\mathscr{B}) \neq \varnothing)$.

$\texttt{CNTBLINTERSPROP}[\mathscr{A}] \leftrightarrow (\forall \mathscr{B})(\subseteq[\mathscr{B}, \mathscr{A}] \rightarrow \texttt{CNTBL}[\mathscr{B}] \rightarrow \neg(\cap(\mathscr{B}) = \varnothing))$

G.27 Section 38

```
DEFINITION MunkTop.38.1: 6-ary relation EQUIVCOMPACTIFICATION. If
COMPACTIFICATION[Y_{1},T_{1},X,T] \wedge
COMPACTIFICATION[Y_{2},T_{2},X,T] then
EQUIVCOMPACTIFICATION[Y_{1},T_{1},Y_{2},T_{2},X,T] \iff
(\exists h : HOMEOMORPHISM[h,Y_{1},T_{1},Y_{2},T_{2}])(
  (\forall x \in X)(h(x) = x)
).
```

DEFINITION MunkTop.38.1: 6-ary relation EQUIVCOMPACTIFICATION.
If $\texttt{COMPACTIFICATION}[Y_1, T_1, X, T] \wedge \texttt{COMPACTIFICATION}[Y_2, T_2, X, T]$ then
$\texttt{EQUIVCOMPACTIFICATION}[Y_1, T_1, Y_2, T_2, X, T] \leftrightarrow (\exists h : \texttt{HOMEOMORPHISM}[h, Y_1, T_1, Y_2, T_2])$
$((\forall x \in X)(h(x) = x))$.

$\texttt{EQUIVCOMPACTIFICATION}[Y_1, T_1, Y_2, T_2, X, T] \leftrightarrow (\texttt{COMPACTIFICATION}[Y_1, T_1, X, T] \wedge$
$\texttt{COMPACTIFICATION}[Y_2, T_2, X, T] \wedge (\exists h)(\texttt{HOMEOMORPHISM}[h, Y_1, T_1, Y_2, T_2] \wedge (\forall x)(x \in$
$X \rightarrow (\iota x_0)(\varpi_0(x, x_0) \in h) = x)))$

```
DEFINITION MunkTop.38.2: 1-ary function Imbedinducedcptfcation. If
TOPIMBED[h,X,T,Z,T'] then Imbedinducedcptfcation(h) \simeq {<Y,T''> :
  COMPACTIFICATION[Y,T'',X,T] \wedge
```

```
  (\exists H : TOPIMBED[H,Y,T'',Z,T'])(
    H EXTENDS h
  )
}.
```

DEFINITION MunkTop.38.2: 1-ary function `Imbedinducedcptfcation`.
If $\texttt{TOPIMBED}[h, X, T, Z, T']$ then $\texttt{Imbedinducedcptfcation}(h) \simeq$
$\{\langle Y, T''\rangle : \texttt{COMPACTIFICATION}[Y, T'', X, T] \wedge (\exists H : \texttt{TOPIMBED}[H, Y, T'', Z, T'])(H\ \texttt{EXTENDS}\ h)\}$.

$\texttt{Imbedinducedcptfcation}(h) \simeq (\iota z_0)(\texttt{TOPIMBED}[h, X, T, Z, T'] \wedge z_0 \simeq (\iota y_0)(\forall x_0)(x_0 \in$
$y_0 \leftrightarrow (\exists Y, T'')(x_0 = \varpi_0(Y, T'') \wedge (\texttt{COMPACTIFICATION}[Y, T'', X, T] \wedge (\exists H)$
$(\texttt{TOPIMBED}[H, Y, T'', Z, T'] \wedge \texttt{EXTENDS}[H, h])))))$

```
DEFINITION MunkTop.38.3: 2-ary function Stonecech.
If COMPLETELYREGULARSP[X,T] then Stonecech(X,T) \simeq
{<Y,T'> :
  COMPACTIFICATION[Y,T',X,T] \wedge
  (\forall f,C,S)(
    COMPACT[C,S] \wedge HAUSDORFF[C,S] \wedge
    CONTINUOUS[f,X,T,C,S] \implies
    (\exists ! g \in Maps(Y,C))(
      CONTINUOUS[g,Y,T',C,S] \wedge
      g EXTENDS f
    )
  )
}.
```

DEFINITION MunkTop.38.3: 2-ary function `Stonecech`. If $\texttt{COMPLETELYREGULARSP}[X, T]$ then $\texttt{Stonecech}(X, T) \simeq \{\langle Y, T'\rangle : \texttt{COMPACTIFICATION}[Y, T', X, T] \wedge (\forall f, C, S)(\texttt{COMPACT}[C, S] \wedge$ $\texttt{HAUSDORFF}[C, S] \wedge \texttt{CONTINUOUS}[f, X, T, C, S] \rightarrow (\exists! g \in \texttt{Maps}(Y, C))(\texttt{CONTINUOUS}[g, Y, T', C, S] \wedge$ $g\ \texttt{EXTENDS}\ f))\}$.

$\texttt{Stonecech}(X, T) \simeq (\iota x_1)(\texttt{COMPLETELYREGULARSP}[X, T] \wedge x_1 \simeq (\iota z_0)(\forall y_0)(y_0 \in z_0 \leftrightarrow$ $(\exists Y, T')(y_0 = \varpi_0(Y, T') \wedge (\texttt{COMPACTIFICATION}[Y, T', X, T] \wedge (\forall f, C, S)\ \texttt{COMPACT}[C, S] \wedge$ $\texttt{HAUSDORFF}[C, S] \wedge \texttt{CONTINUOUS}[f, X, T, C, S] \rightarrow (\exists x_0)(\forall g)(g \in \texttt{Maps}(Y, C) \wedge$ $\texttt{CONTINUOUS}[g, Y, T', C, S] \wedge \texttt{EXTENDS}[g, f] \leftrightarrow g = x_0)))))$

Bibliography

[1] Mizar Home Page: http://mizar.org.

[2] Wikipedia: http://en.wikipedia.org/wiki/LL_parser.

[3] Alfred V. Aho and Jeffrey D. Ullman. *The Theory of Parsing, Translation, and Compiling*, volume 1. Prentice-Hall, Englewood Cliffs, N.J., 1972.

[4] Andrea Asperti, Grzegorz Bancerek, and Andrzej Trybulec, editors. *Mathematical Knowledge Management: Third International Conference, MKM 2004*, Lecture Notes in Computer Science. Springer, 2004.

[5] Jeremy Avigad. Personal communication.

[6] Jeremy Avigad. *Understanding Proofs*. Oxford University Press.

[7] Arnon Avron. Formalizing set theory as it is actually used. In Asperti et al. [4], pages 32–43.

[8] Solomon Feferman. Definedness. U.C. Irvine, May 1995.

[9] H. Friedman and R. C. Flagg. A framework for measuring the complexity of mathematical concepts. *Advances in Applied Mathematics*, 11:1–34, 1990.

[10] Harvey M. Friedman. The formalization of mathematics. Available online: http://www.math.ohio-state.edu/~friedman/, May 1997.

[11] Harvey M. Friedman. Proofless text. September 2005.

[12] James R. Munkres. *Topology*. Prentice Hall, Upper Saddle River, N.J., second edition.

[13] Joe Ramsey. Personal communication.

[14] Frdrich Wilhelm Schröer. Accent, a compiler compiler for the entire class of context-free grammars. Technical report, Fraunhofer Institute for Computer Architecture and Software Technology, 2006.

[15] Patrick Suppes. *Axiomatic Set Theory*. Van Nostrand, Princeton, 1960.

[16] A.S. Troelstra and D. van Dalen. *Constructivism in mathematics: an introduction*, volume 1. North-Holland, Amsterdam, 1988.