# A gentle introduction to map theory

Klaus Grue

January 2, 2007

## Contents

**1 Introduction** **2**
1.1 ZFC . . . 2
1.2 Overview . . . 3
1.3 History . . . 3

**2 The M programming language** **4**
2.1 Syntax . . . 4
2.2 The syntax expressed in ZFC . . . 4
2.3 Semantics . . . 5
2.4 Reduction steps . . . 5
2.5 Branch . . . 6
2.6 Application . . . 6
2.7 Parallel disjunction . . . 6
2.8 Existence . . . 7

**3 A theory of M** **7**
3.1 The M-computer . . . 7
3.2 Equivalence . . . 8
3.3 Elementary properties . . . 8
3.4 Quartum non datur . . . 9
3.5 The Scott order . . . 10
3.6 Extensionality . . . 11

**4 Models** **11**
4.1 Trees . . . 11
4.2 Compact maps . . . 12
4.3 Prime maps . . . 13
4.4 Coherent maps . . . 14
4.5 Model building kit . . . 14
4.6 Model building . . . 15

**5 Quantification** **15**
5.1 Choices to be made . . . . . 15
5.2 Choice of quantifier . . . . . 16
5.3 New function letters . . . . . 16
5.4 Problems with $\varepsilon$ . . . . . 17
5.5 Strict versus lazy quantifiers . . . . . 17
5.6 The syntax of Map Theory . . . . . 18
5.7 Axioms of quantification . . . . . 19

**6 Choice of domain** **19**
6.1 Size of domain . . . . . 19
6.2 Representation of sets . . . . . 20
6.3 Wellfoundedness . . . . . 20
6.4 Continuity . . . . . 21
6.5 $\kappa$-Scott Continuity . . . . . 22
6.6 Uniform continuity . . . . . 22
6.7 A property of $\Phi$ . . . . . 23
6.8 Classes . . . . . 23
6.9 The dual class . . . . . 23
6.10 The definition of $\phi$ . . . . . 24

**7 Conclusion** **24**

**Abstract**

Map theory (MT) is a foundation of mathematics. MT is slightly more powerful than ZFC set theory, but its merit is that it builds on top of computable functions where ZFC builds on top of finite sets. Thus, where ZFC is suited as a foundation of mathematics, MT is suited as a foundation of both mathematics and computer science. This paper gives an introduction to MT intended for a broad audience consisting of mathematicians, logicians, and computer scientists.

# 1 Introduction

## 1.1 ZFC

Recall that ZFC is a generalization of the theory of finite sets:

- The power set of a finite set is finite. Therefore, ZFC has a power set operator.
- The union of a finite set of finite sets is finite. Therefore, ZFC has a union operator.
- The complement of a finite set is not finite. Therefore, ZFC has no complement operator.

Even the axiom of choice, which is also known as the axiom of Zermelo, is valid in the world of finite sets. The only axiom of ZFC which transcends the world of finite sets is the axiom of infinity.

Map theory is constructed the same way, except that it begins with computable functions.

## 1.2 Overview

We introduce map theory in the following steps:

**Section 2** defines a programming language M for expressing computable functions. The notion of computable functions is much richer than that of finite sets, and for that reason we shall dwell longer on computable functions in MT than it is usual to dwell on finite sets in ZFC.

**Section 3** presents a theory of M, i.e. a theory which describes the behavior of programs expressed in the M programming language. Section 3 also defines a relation of behavioral equivalence of programs. Programs modulo this equivalence relation are referred to as maps.

**Section 4** presents a mental model for visualizing maps. That mental model can be extended into a firm, mathematical model, but we shall not do so in the present paper.

**Section 5** discusses which quantifiers can be added to the M programming language. Section 5 also states the four axioms which turn the theory from Section 3 into Map Theory. The axioms do not complete the definition of MT, however, since they refer to a construct $\phi$ which is defined in Section 6.

**Section 6** selects the domain of quantification. It turns out to be a bad idea to let the quantifiers of Map Theory quantify over all maps. Rather, Section 6 defines the notion of a 'wellfounded' map and lets quantifiers quantify over these maps. Section 6 also defines a predicate $\phi$ for testing whether or not a map is wellfounded.

MT is a foundation which can stand on its own feet. But we utilize the readers previous knowledge and use ZFC as a quick way of explaining MT.

## 1.3 History

MT was introduced in 1992 in [3]. [3] proved all axioms and inference rules of ZFC in MT and also proved the consistency of MT in ZFC+SI where SI is the assumption that there exists an inaccessible ordinal.

In 1997, [1] showed that MT has a quite natural Scott model. That made MT much easier to understand.

MT from 1992 has a peculiar, ad hoc list of 'construction' axioms. A version where all the construction axioms are replaced by a definition was published in

2002 [4], but the consistency has never been proved and a model of the version in [4] is likely to be much less natural than the one in [1].

The present paper proposes a new axiomatization. This new version still replaces the construction axioms by a definition. But the model in [1] is expected to satisfy the new version. Furthermore, the new version is expected to be able to prove all axioms and inference rules of ZFC. Verification of the last two claims, however, is future work.

# 2 The M programming language

## 2.1 Syntax

The programming language M underlying MT has the following BNF syntax:[1]

$$
\begin{array}{lcll}
\mathcal{V} & ::= & \mathsf{v}_1 \mid \mathsf{v}_2 \mid \mathsf{v}_3 \mid \cdots & \text{(object Variable)} \\
\mathcal{U} & ::= & \mathsf{T} & \text{(Uhr element)} \\
\mathcal{F} & ::= & \lambda\mathcal{V}.\mathcal{T} & \text{(Function)} \\
\mathcal{N} & ::= & \mathcal{U} \mid \mathcal{F} & \text{(Normal form)} \\
\mathcal{A} & ::= & \mathcal{T}\mathcal{T} & \text{(functional Application)} \\
\mathcal{B} & ::= & \mathsf{if}(\mathcal{T},\mathcal{T},\mathcal{T}) & \text{(Branch)} \\
\mathcal{D} & ::= & \mathcal{T}\|\mathcal{T} & \text{(parallel Disjunction)} \\
\mathcal{E} & ::= & \mathsf{E}\mathcal{T} & \text{(pure Existence)} \\
\mathcal{R} & ::= & \mathcal{A} \mid \mathcal{B} \mid \mathcal{D} \mid \mathcal{E} & \text{(Reducible term)} \\
\mathcal{T} & ::= & \mathcal{V} \mid \mathcal{N} \mid \mathcal{R} & \text{(Term)}
\end{array}
$$

As an example, $\lambda\mathsf{v}_1.\mathsf{if}(\mathsf{v}_1,\mathsf{v}_2,\mathsf{T})$ is a term:

$$\lambda\mathsf{v}_1.\mathsf{if}(\mathsf{v}_1,\mathsf{v}_2,\mathsf{T}) \quad \in \quad \mathcal{T}$$

Let $\mathcal{C}$ denote the set of closed terms (i.e. terms without free variables). As examples, we have

$$
\begin{array}{ll}
\lambda\mathsf{v}_1.\mathsf{v}_1 & \in \quad \mathcal{C} \\
\lambda\mathsf{v}_1.\mathsf{v}_2 & \notin \quad \mathcal{C}
\end{array}
$$

## 2.2 The syntax expressed in ZFC

We now translate the BNF-definition of the previous section into ZFC. Define

$$
\begin{array}{lcl}
\mathsf{v}_i & \doteq & \langle 0, i\rangle \\
\mathsf{T} & \doteq & \langle 1\rangle \\
\lambda x.y & \doteq & \langle 2, x, y\rangle \\
xy & \doteq & \langle 3, x, y\rangle \\
\mathsf{if}(x,y,z) & \doteq & \langle 4, x, y, z\rangle \\
x\|y & \doteq & \langle 5, x, y\rangle \\
\mathsf{E}x & \doteq & \langle 6, x\rangle
\end{array}
$$

[1] Readers with a $\lambda$-calculus background should note that M is an impure $\lambda$-calculus because of the uhr-element and that the definition of 'normal form' is highly non-standard.

Let $\omega$ be the set of finite ordinals (i.e. the set of natural numbers). Let $\mathcal{V}$, $\mathcal{U}$, $\mathcal{F}$, $\mathcal{N}$, $\mathcal{A}$, $\mathcal{B}$, $\mathcal{D}$, $\mathcal{E}$, $\mathcal{R}$, and $\mathcal{T}$ be the smallest sets such that

$$\begin{array}{lcll}
i \in \omega & \Rightarrow & \mathsf{v}_i & \in \mathcal{V} \quad \text{(object Variable)} \\
x = \mathsf{T} & \Rightarrow & x & \in \mathcal{U} \quad \text{(Uhr element)} \\
x \in \mathcal{V} \wedge y \in \mathcal{T} & \Rightarrow & \lambda x.y & \in \mathcal{F} \quad \text{(Function)} \\
x \in \mathcal{U} \cup \mathcal{F} & \Rightarrow & x & \in \mathcal{N} \quad \text{(Normal form)} \\
x, y \in \mathcal{T} & \Rightarrow & xy & \in \mathcal{A} \quad \text{(functional Application)} \\
x, y, z \in \mathcal{T} & \Rightarrow & \mathsf{if}(x, y, z) & \in \mathcal{B} \quad \text{(Branch)} \\
x, y \in \mathcal{T} & \Rightarrow & x \| y & \in \mathcal{D} \quad \text{(Parallel disjunction)} \\
x \in \mathcal{T} & \Rightarrow & \mathsf{E}x & \in \mathcal{E} \quad \text{(pure Existence)} \\
x \in \mathcal{A} \cup \mathcal{B} \cup \mathcal{D} \cup \mathcal{E} & \Rightarrow & x & \in \mathcal{R} \quad \text{(Reducible term)} \\
x \in \mathcal{N} \cup \mathcal{R} & \Rightarrow & x & \in \mathcal{T} \quad \text{(Term)}
\end{array}$$

## 2.3 Semantics

The semantics of M is a mathematical description of what a computer is supposed to do with terms $t$.

The semantics of M only considers closed terms $t \in \mathcal{C}$. Given a closed term $t$, a computer is supposed to *reduce* $t$, i.e. to transform $t$ according to certain *reduction rules* until $t$ is transformed into normal form, if possible. Hence, the semantics of M is a function $f_*$ of type

$$f_* : \mathcal{C} \to \mathcal{N}$$

where $f_*(t)$ is defined iff $t$ can be reduced to normal form. The function $f_*$ is *partial* in the sense that it is not defined for all $t \in \mathcal{C}$.

## 2.4 Reduction steps

A computer is supposed to reduce $t$ one *step* at a time, i.e. to compute a *reduction sequence*

$$t_0 \to t_1 \to t_2 \to \cdots$$

of closed terms such that $t_0 = t$ and $t_{i+1} = f_1(t_i)$ where $f_1 : \mathcal{C} \to \mathcal{C}$ defines what to do in one step. If $t_i \in \mathcal{N}$ for no $i$ then $f_*(t)$ is undefined. If $t_i \in \mathcal{N}$ for some $i$ then $f_*(t) = t_i$. The definition of $f_1(t)$ reads:

$$f_1(t) = \begin{cases} t & \text{if } t \in \mathcal{N} \\ f_{\mathcal{A}}(t) & \text{if } t \in \mathcal{A} \\ f_{\mathcal{B}}(t) & \text{if } t \in \mathcal{B} \\ f_{\mathcal{D}}(t) & \text{if } t \in \mathcal{D} \\ f_{\mathcal{E}}(t) & \text{if } t \in \mathcal{E} \end{cases}$$

The function $f_1$ is *total* in the sense that $f_1(t)$ is defined for all $t \in \mathcal{C}$.

As an example, the reduction sequence of $\mathsf{T}$ is

$$\mathsf{T} \to \mathsf{T} \to \mathsf{T} \to \cdots$$

so $f_*(\mathsf{T}) = \mathsf{T}$.

## 2.5 Branch

The function $f_{\mathcal{B}}$ is defined as follows for all $x, y, z \in \mathcal{C}$:

$$f_{\mathcal{B}}(\mathsf{if}(x,y,z)) = \begin{cases} y & \text{if } x \in \mathcal{U} \\ z & \text{if } x \in \mathcal{F} \\ \mathsf{if}(f_1(x),y,z) & \text{if } x \in \mathcal{R} \end{cases}$$

As an example, the reduction sequence of $\mathsf{if}(\mathsf{if}(\mathsf{T},\mathsf{T},\mathsf{T}),\lambda\mathsf{v}_1.\mathsf{v}_1,\mathsf{T})$ is

$$\mathsf{if}(\mathsf{if}(\mathsf{T},\mathsf{T},\mathsf{T}),\lambda\mathsf{v}_1.\mathsf{v}_1,\mathsf{T}) \to \mathsf{if}(\mathsf{T},\lambda\mathsf{v}_1.\mathsf{v}_1,\mathsf{T}) \to \lambda\mathsf{v}_1.\mathsf{v}_1 \to \lambda\mathsf{v}_1.\mathsf{v}_1 \to \cdots$$

so $f_*(\mathsf{if}(\mathsf{if}(\mathsf{T},\mathsf{T},\mathsf{T}),\lambda\mathsf{v}_1.\mathsf{v}_1,\mathsf{T})) = \lambda\mathsf{v}_1.\mathsf{v}_1$.

## 2.6 Application

For all $a, b \in \mathcal{T}$ and $v \in \mathcal{V}$ define $\langle\, a \,|v{:=}\, b \,\rangle$ as the result of replacing all free occurrences of $v$ in $a$ by $b$ with suitable renaming of bound variables to avoid variable clashes. We shall assume that renaming is done in some deterministic way which we shall not specify any further. Now define apply: $\mathcal{F} \times \mathcal{T} \to \mathcal{T}$ such that

$$\text{apply}(\lambda v.a, b) = \langle\, a \,|v{:=}\, b \,\rangle$$

The function $f_{\mathcal{A}}$ is defined as follows for all $x, y \in \mathcal{C}$:

$$f_{\mathcal{A}}(xy) = \begin{cases} x & \text{if } x \in \mathcal{U} \\ \text{apply}(x,y) & \text{if } x \in \mathcal{F} \\ f_1(x)y & \text{if } x \in \mathcal{R} \end{cases}$$

As an example, the reduction sequence of $(\lambda\mathsf{v}_1.\mathsf{v}_1\mathsf{v}_1)(\lambda\mathsf{v}_1.\mathsf{v}_1\mathsf{v}_1)$ is

$$(\lambda\mathsf{v}_1.\mathsf{v}_1\mathsf{v}_1)(\lambda\mathsf{v}_1.\mathsf{v}_1\mathsf{v}_1) \to (\lambda\mathsf{v}_1.\mathsf{v}_1\mathsf{v}_1)(\lambda\mathsf{v}_1.\mathsf{v}_1\mathsf{v}_1) \to \cdots$$

so $f_*((\lambda\mathsf{v}_1.\mathsf{v}_1\mathsf{v}_1)(\lambda\mathsf{v}_1.\mathsf{v}_1\mathsf{v}_1))$ is undefined.

## 2.7 Parallel disjunction

The function $f_{\mathcal{D}}$ is defined as follows for all $x, y \in \mathcal{C}$:

$$f_{\mathcal{D}}(x\|y) = \begin{cases} \mathsf{T} & \text{if } x \in \mathcal{U} \text{ or } y \in \mathcal{U} \\ \lambda\mathsf{v}_1.(x\mathsf{v}_1)\|(y\mathsf{v}_1) & \text{if } x \in \mathcal{F} \text{ and } y \in \mathcal{F} \\ f_1(x)\|f_1(y) & \text{otherwise} \end{cases}$$

Hence, $x\|y$ is reduced by reducing $x$ and $y$ in parallel until one of them reduces to $\mathsf{T}$ or both of them reduce to functions.

### 2.8 Existence

The function $f_{\mathcal{E}}$ is defined as follows for all $x \in \mathcal{C}$:

$$\begin{array}{lcl} S & \doteq & \lambda \mathsf{v}_1.\lambda \mathsf{v}_2.\lambda \mathsf{v}_3.\mathsf{v}_1\mathsf{v}_3(\mathsf{v}_2\mathsf{v}_3) \\ K & \doteq & \lambda \mathsf{v}_1.\lambda \mathsf{v}_2.\mathsf{v}_1 \\ B & \doteq & \lambda \mathsf{v}_1.\lambda \mathsf{v}_2.\lambda \mathsf{v}_3.\mathsf{if}(\mathsf{v}_1,\mathsf{v}_2,\mathsf{v}_3) \\ D & \doteq & \lambda \mathsf{v}_1.\lambda \mathsf{v}_2.\mathsf{v}_1\|\mathsf{v}_2 \\ E & \doteq & \lambda \mathsf{v}_1.\mathsf{E}\mathsf{v}_1 \\ A & \doteq & \lambda \mathsf{v}_1.\mathsf{E}\lambda \mathsf{v}_2.\mathsf{E}\lambda \mathsf{v}_3.\mathsf{v}_1(\mathsf{v}_2\mathsf{v}_3) \\ f_{\mathcal{E}}(\mathsf{E}x) & = & xS\|xK\|x\mathsf{T}\|xB\|xD\|xE\|Ax \end{array}$$

Essentially, $\mathsf{E}x$ is reduced by reducing $xy$ for all $y \in \mathcal{C}$ in parallel. $\mathsf{E}x$ reduces to $\mathsf{T}$ iff $xy$ reduces to $\mathsf{T}$ for some $y \in \mathcal{C}$. $\mathsf{E}x$ is a very weak quantifier. At a later stage, map theory is constructed by adding a different and much stronger quantifier.

## 3 A theory of M

### 3.1 The M-computer

One may think of an implementation of the M programming language as an *M-computer* with two lamps and a keyboard. The two lamps are labeled $\mathsf{T}$ and $\mathcal{F}$, and the keyboard has the symbols used in the BNF-definition in Section 2.1. When a user enters a closed term on the keyboard, the computer starts reducing the term. If the term reduces to a normal form, then the $\mathsf{T}$ and $\mathcal{F}$ lamp lights if the result is in $\mathcal{U}$ and $\mathcal{F}$, respectively[2].

We shall say that a closed term $t$ is a *true*, *function*, or *bottom* term if $f(t) \in \mathcal{U}$, $f(t) \in \mathcal{F}$, or $f(t)$ is undefined, respectively. Hence, if the M-computer receives a true term, it lights the $\mathsf{T}$ lamp, if it receives a function term, it lights the $\mathcal{F}$ lamp, and if it receives a bottom term then it will work indefinitely without lighting either lamp.

Hence, an ordinary user of the M-computer can use it to verify that a term is a true or a function term. It takes a clairvoyant user to verify that a term is a bottom term.

We shall say that two closed terms $r$ and $s$ are *root equivalent*, written $r \approx s$, if they are both true, both function, or both bottom terms. Root equivalence is not computable because it is undecidable whether or not a term is a bottom term.

[2]Readers with a background in $\lambda$-calculus will note that this is very different from a typical implementation of pure lambda calculus: in pure $\lambda$-calculus, the result of a computation is a normal form; in M, the result of a computation is binary and the user cannot access the normal form

## 3.2 Equivalence

We shall say that two closed terms $r$ and $s$ are *equivalent*, written $r \equiv s$, if $tr \approx ts$ for all closed terms $t$. Hence, two terms are equivalent if they behave the same on the M-computer in any context $t$.

From the point of view of MT, $r \equiv s$ expresses equality.

In the present paper, we use $r = s$ to denote equality in ZFC. Hence, for closed terms $r$ and $s$, $r = s$ expresses that $r$ and $s$ are the same term and $r \equiv s$ expresses that $r$ and $s$ are equal in MT. In the present paper, $r \equiv s$ is an equivalence relation in ZFC.

Some papers on MT is based on MT rather than ZFC. In those papers, equality in MT is still written $x \equiv y$ whereas $x = y$ is used for quite another purpose: to denote equality modulo identifications. So in those papers, $x \equiv y$ is equality whereas $x = y$ merely is an equivalence relation.

We shall refer to the equivalence classes of the class division $\mathcal{C}/\equiv$ as *maps*[3].

## 3.3 Elementary properties

Let $x \propto \langle y|v{:=}z\rangle$ denote that $x$ is identical to $\langle\, y\,|v{:=}\,z\,\rangle$ except for renaming of bound variables and define

$$\begin{array}{lcl} \bot & \doteq & (\lambda \mathsf{v}_1.\mathsf{v}_1\mathsf{v}_1)(\lambda \mathsf{v}_1.\mathsf{v}_1\mathsf{v}_1) \\ x \circ y & \doteq & (\lambda \mathsf{v}_1.\lambda \mathsf{v}_2.\lambda \mathsf{v}_3.\mathsf{v}_1(\mathsf{v}_2\mathsf{v}_3))xy \\ ? & \doteq & \lambda \mathsf{v}_1.\mathsf{if}(\mathsf{v}_1, \mathsf{T}, \bot) \\ x \to y & \Leftrightarrow & \mathsf{if}(x, y, \mathsf{T}) \equiv \mathsf{if}(x, \mathsf{T}, \mathsf{T}) \end{array}$$

We have that $\bot$ is a bottom term, $x \circ y$ is functional composition, ? is a function which maps $\mathsf{T}$ to $\mathsf{T}$ and anything else to $\bot$, and $x \to y$ is one way to express that $x \equiv \mathsf{T}$ implies $y \equiv \mathsf{T}$ in MT. We state without proof that the following

[3]So the set of maps is countable until further. The notion of a map is going to be generalized when we introduce MT.

hold for all terms $x, y, z \in \mathcal{T}$ and all variables $v \in \mathcal{V}$:

$$x \equiv y \Rightarrow x \equiv z \Rightarrow y \equiv z^4$$
$$x \equiv y \Rightarrow \lambda v.x \equiv \lambda v.y$$
$$x \equiv y \Rightarrow zx \equiv zy$$

$$\mathsf{if}(\mathsf{T}, y, z) \equiv y$$
$$\mathsf{if}(\lambda v.x, y, z) \equiv z$$
$$\mathsf{if}(\bot, y, z) \equiv z$$

$$\mathsf{T}y \equiv \mathsf{T}$$
$$(\lambda v.y)z \equiv x \quad \text{if } x \propto \langle y|v{:=}z\rangle$$
$$\bot y \equiv \bot$$

$$\mathsf{T}\|x \equiv \mathsf{T}$$
$$x\|\mathsf{T} \equiv \mathsf{T}$$
$$(\lambda v.x)\|(\lambda v.y) \equiv \lambda v.(x\|y)$$
$$(\lambda v.x)\|\bot \equiv \bot$$
$$\bot\|(\lambda v.x) \equiv \bot$$
$$\bot\|\bot \equiv \bot$$

$$\mathsf{E}\mathsf{T} \equiv \mathsf{T}$$
$$\mathsf{E}\bot \equiv \bot$$
$$\mathsf{E}(x \circ y) \to \mathsf{E}x$$
$$\mathsf{E}(? \circ x) \equiv \mathsf{E}x$$

The statements above are the elementary axioms and inference rules of MT.

### 3.4 Quartum non datur

The rule of Quartum Non Datur (QND) reads:

$$x\mathsf{T} \equiv y\mathsf{T} \Rightarrow x(\lambda \mathsf{v}_1.\mathsf{v}_2\mathsf{v}_1) \equiv y(\lambda \mathsf{v}_1.\mathsf{v}_2\mathsf{v}_1) \Rightarrow x\bot \equiv y\bot \Rightarrow x\mathsf{v}_2 \equiv y\mathsf{v}_2$$

The rule says that any closed term is a true, function, or bottom term, there is no fourth possibility. QND allows to prove a lemma by cases. Now define:

$$\begin{array}{lcl} \mathsf{F} & \doteq & \lambda \mathsf{v}_1.\mathsf{T} \\ \neg x & \doteq & \mathsf{if}(x, \mathsf{F}, \mathsf{T}) \\ x \wedge y & \doteq & \mathsf{if}(x, \mathsf{if}(y, \mathsf{T}, \mathsf{F}), \mathsf{if}(y, \mathsf{F}, \mathsf{F})) \end{array}$$

We use $\mathsf{F}$ to represent falsehood and $\neg x$ and $x \wedge y$ express negation and conjunction in M (and, thereby, in MT). We use $\wedge$ for conjunction in both MT and ZFC, but which one is meant should be clear from the context. As an example, in the fact

$$(x \wedge y) \equiv \mathsf{T} \Leftrightarrow (x \equiv \mathsf{T}) \wedge (y \equiv \mathsf{T})$$

[4] We take $\Rightarrow$ to be right associative so that $A \Rightarrow B \Rightarrow C$ means $A \Rightarrow (B \Rightarrow C)$ which is equivalent to $A \wedge B \Rightarrow C$.

the leftmost $\wedge$ is part of an MT term whereas the rightmost $\wedge$ is part of a ZFC formula. Note, by the way, that conjunction lives at the level of terms in MT and at the level of formulas in ZFC. As we shall see later, the same holds true for quantifiers. The elementary properties listed in Section 3.3 allow to prove e.g.

$$\left|\begin{array}{lll} \neg\mathsf{T} & \equiv & \mathsf{F} \\ \neg\mathsf{F} & \equiv & \mathsf{T} \\ \neg\bot & \equiv & \bot \end{array}\right| \left|\begin{array}{lll} \mathsf{T}\wedge\mathsf{T} & \equiv & \mathsf{T} \\ \mathsf{F}\wedge\mathsf{T} & \equiv & \mathsf{F} \\ \bot\wedge\mathsf{T} & \equiv & \bot \end{array}\right| \left|\begin{array}{lll} \mathsf{T}\wedge\mathsf{F} & \equiv & \mathsf{F} \\ \mathsf{F}\wedge\mathsf{F} & \equiv & \mathsf{F} \\ \bot\wedge\mathsf{F} & \equiv & \bot \end{array}\right| \left|\begin{array}{lll} \mathsf{T}\wedge\bot & \equiv & \bot \\ \mathsf{F}\wedge\bot & \equiv & \bot \\ \bot\wedge\bot & \equiv & \bot \end{array}\right|$$

QND is required to prove more general statements like $\mathsf{v}_1 \wedge \mathsf{v}_2 \equiv \mathsf{v}_2 \wedge \mathsf{v}_1$ and $\neg\neg\neg\mathsf{v}_1 \equiv \neg\mathsf{v}_1$. QND is unable to prove $\neg\neg\mathsf{v}_1 \equiv \mathsf{v}_1$ for the simple reason that $\neg\neg\mathsf{v}_1 \equiv \mathsf{v}_1$ does not hold in general (as an example, $\neg\neg\lambda\mathsf{v}_1.\bot \equiv \neg\mathsf{T} \equiv \mathsf{F} \equiv \lambda\mathsf{v}_1.\mathsf{T} \not\equiv \lambda\mathsf{v}_1.\bot$).

## 3.5 The Scott order

Now define

$$\begin{array}{lcl} \mathsf{Y} & \doteq & \lambda\mathsf{v}_1.(\lambda\mathsf{v}_2.\mathsf{v}_1(\mathsf{v}_2\mathsf{v}_2))(\lambda\mathsf{v}_2.\mathsf{v}_1(\mathsf{v}_2\mathsf{v}_2)) \\ x \downarrow y & \doteq & \mathsf{if}(x, \mathsf{if}(y, \mathsf{T}, \bot), \mathsf{if}(y, \bot, \lambda z.xz \downarrow yz)) \\ x \preceq y & \Leftrightarrow & x \equiv x \downarrow y \end{array}$$

The elementary properties allow to prove $\mathsf{Y}x \equiv x(\mathsf{Y}x)$ showing that $\mathsf{Y}$ is a fixed point operator. Having a fixed point operator makes recursive definitions like the second definition above permissible as any recursive definition can be translated to a non-recursive one using $\mathsf{Y}$.

In the 'standard model' of MT, $x \downarrow y$ is the infimum (greatest lower bound) of $x$ and $y$ w.r.t. the Scott order whereas $x \preceq y$ expresses the Scott order itself.

Having defined the Scott order $x \preceq y$ we can state two more rules about maps, one which says that all maps are monotonic, and one which says that $\mathsf{Y}$ produces minimal fixed points:

$$\begin{array}{l} x \preceq y \Rightarrow zx \preceq zy \\ xy \preceq y \Rightarrow \mathsf{Y}x \preceq y \end{array}$$

In addition, M also satisfies the property of Scott continuity, but that property is not included as an inference rule of MT since that would hinder the introduction of genuine quantifiers (the $\mathsf{E}x$ quantifier is not a genuine quantifier; it can only return $\mathsf{T}$ and $\bot$ but not $\mathsf{F}$).

As proved in [1], full map theory has a generalized property known as $\kappa$-Scott continuity where $\kappa$ is a suitably chosen ordinal. As a special case of $\kappa$-Scott continuity, ordinary Scott continuity is identical to $\omega$-Scott continuity where $\omega$ is the set of finite ordinals.

### 3.6 Extensionality

ZFC has the *extensionality* property that the following are equivalent:

1. $x \in z \Leftrightarrow y \in z$ for all sets $z$.
2. $z \in x \Leftrightarrow z \in y$ for all sets $z$.

M happens to have a similar property which says that the following slightly more complicated statements are equivalent:

1. $zx \approx zy$ for all $z \in \mathcal{C}$.
2. $xz_1 \cdots z_n \approx yz_1 \cdots z_n$ for all $n \in \mathbf{N}$ and all $z_1, \ldots, z_n \in \mathcal{C}$.

Statement (1) above is the definition of $x \equiv y$ and (2) follows from (1) according to the elementary properties of M. Even though it may not be evident, the following *rule of extensionality* expresses that (1) follows from (2):

$$\neg\neg(xu) \equiv \neg\neg(yu) \Rightarrow xuv \equiv x(zuv) \Rightarrow yuv \equiv y(zuv) \Rightarrow xu \equiv yu$$

Above, $u, v \in \mathcal{V}$ must be distinct variables and $x, y, z \in \mathcal{T}$ must be terms in which $u$ and $v$ do not occur free.

Extensionality allows to prove $x \downarrow x \equiv x$, $x \downarrow y \equiv y \downarrow x$, and $x \downarrow (y \downarrow z) \equiv (x \downarrow y) \downarrow z$ (c.f. [5]). These results in turn allow to prove $x \preceq x$, $x \preceq y \Rightarrow y \preceq x \Rightarrow x \equiv y$, and $x \preceq y \Rightarrow y \preceq z \Rightarrow x \preceq z$.

# 4 Models

## 4.1 Trees

As mentioned in Section 3.6, two maps $x$ and $y$ are equal iff

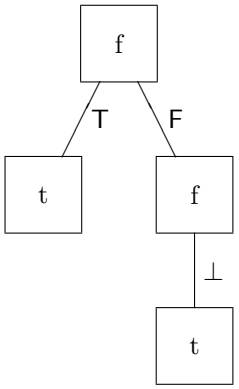
$$xz_1 \cdots z_n \approx yz_1 \cdots z_n \quad \text{for all } n \in \mathbf{N} \text{ and all } z_1, \ldots, z_n \in \mathcal{C}$$

That property may be used as basis for a useful mental picture of what a map looks like. The mental picture described in the following is a graphical one which we illustrate by 'drawing' the map $\mathsf{I} = \lambda x.x$.

Among other, the map $\mathsf{I}$ has the following properties:

$$\begin{array}{lll} \mathsf{I} & \in & \mathcal{F} \\ \mathsf{IT} & \in & \mathcal{U} \\ \mathsf{IF} & \in & \mathcal{F} \\ \mathsf{IF}\bot & \in & \mathcal{U} \end{array}$$

The information above may be represented by the following graphical construction:




In the drawing, nodes labeled t and f represent the sets $\mathcal{U}$ and $\mathcal{F}$, respectively. The f in the root represents the $\mathcal{F}$ in $\mathsf{I} \in \mathcal{F}$. The t in the bottom right corner represents the $\mathcal{U}$ in $\mathsf{IF}\bot \in \mathcal{U}$.

In general, in a drawing of a map $m$, if $mx_1 \cdots x_n \in \mathcal{F}$ then there is a node labeled f and a path from the root to that node labeled $x_1, \ldots, x_n$.

We do not allow downward edges from nodes labeled t, so the rule for nodes labeled t is a bit more complicated: If $mx_1 \cdots x_n \in \mathcal{U}$ and if $mx_1 \cdots x_m \notin \mathcal{U}$ for all $m < n$ then there is a node labeled t and a path from the root to that node labeled $x_1, \ldots, x_n$.

A drawing like the one above only records positive information of form $mx_1 \cdots x_n \in \mathcal{U}$ or $mx_1 \cdots x_n \in \mathcal{F}$. It does not record negative information of form $mx_1 \cdots x_n \equiv \bot$.

The drawing above merely is an approximation of $\mathsf{I}$ since it does not record all paths $x_1 \cdots x_n$ for which $\mathsf{I}x_1 \cdots x_n \not\equiv \bot$. The picture above may be seen as a drawing of

$$\lambda x.\mathsf{if}(x\bot, \mathsf{T}, \lambda y.\mathsf{T})$$

since that is the smallest map w.r.t. the Scott order which has the illustrated properties. That map is indeed an *approximation* of $\mathsf{I}$ in the sense that

$$\lambda x.\mathsf{if}(x\bot, \mathsf{T}, \lambda y.\mathsf{T}) \preceq \mathsf{I}$$

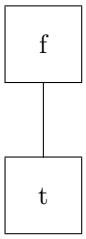
A full drawing of $\mathsf{I}$ would be infinitely large.

## 4.2 Compact maps

We shall say that a map $c$ is *compact* iff there exists a map $\chi_c$ such that $\chi_c d \equiv \mathsf{T}$ iff $c \preceq d$ for all maps $d$. The compact maps are exactly those which are also compact in the standard model of map theory.

In drawings, one may restrict edge labels to compact maps without loss of information. From now on, we assume that edge labels are restricted to be compact maps.

In the drawing of I in Section 4.1, the edge labels T, F, and $\perp$ are themselves maps and may themselves be drawn instead of referred to. A drawing of T consists of a single node labeled t. A drawing of $\perp$ is hard to see since it consists of a tree with no nodes. A drawing of F may look thus:

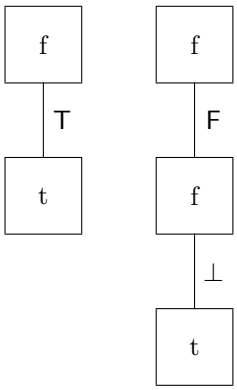



The drawing consists of an f-node and a t-node connected by an edge labeled $\perp$. The $\perp$ can (or, rather, cannot) be seen next to the label.

All compact maps of M happen to have finitely large drawings[5]. This holds true even if one insists that all edge labels have to be drawn themselves. Compact maps of full MT need not have finitely large drawings.

## 4.3 Prime maps

The map drawn in Section 4.1 may be seen as the least upper bound of the following two maps:




We shall say that a map is *prime* if it has finitely many nodes and each node has at most one downward edge. The prime maps are exactly those which are also prime in the standard model of map theory. As a technicality we require prime maps to have at least one node so that we expel $\perp$ from the society of prime maps. The two maps above constitute a prime factoring of the map drawn in Section 4.1.

[5]More precisely: for all compact maps $c$ there exists a finitely large drawing which represents $c$. Drawings may contain redundant information and a compact map can easily have many finite as well as many infinite drawings.

## 4.4 Coherent maps

We shall say that two maps $x$ and $y$ are *coherent*, written $x \bigcirc y$, if they have an upper bound, i.e. if there exists a map $z$ such that $x \preceq z$ and $y \preceq z$. The two prime maps in Section 4.3 are coherent.

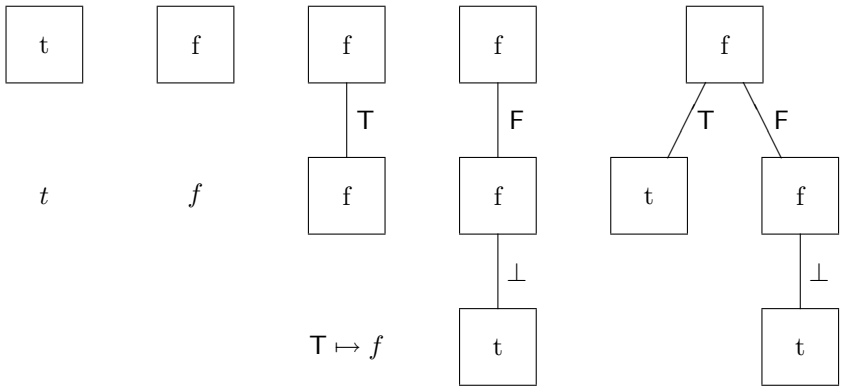
If two maps $x$ and $y$ have a common upper bound $z$ then they also have a least upper bound $\mathrm{lub}(x, y, z)$ where

$$\begin{array}{lcl} \mathrm{lub}(x, y, z) & \doteq & \mathsf{if}(z, x : y : \mathsf{T}, \neg x : \neg y : \lambda u.\mathrm{lub}(xu, yu, zu)) \\ x : y & \doteq & \mathsf{if}(x, y, \bot) \end{array}$$

For coherent maps $x$ and $y$, let $x \uparrow y$ denote the least upper bound of $x$ and $y$. For any finite set $S$ of pairwise coherent maps let $\uparrow S$ denote the least upper bound of all the maps in $S$.[6]

## 4.5 Model building kit

For compact $c$ and maps $m$ let $c \mapsto m$ denote $\lambda x.\mathsf{if}(\chi_c x, m, \bot)$. Furthermore, let $t$ and $f$ denote $\mathsf{T}$ and $\lambda x.\bot$, respectively. A number of drawings illustrate the capabilities of $t$, $f$, $\uparrow$, and $\mapsto$:




We have, $t = \mathsf{T}$, $\uparrow\emptyset = \bot$, and $(\uparrow\emptyset) \mapsto t = \mathsf{F}$, so we may also express the rightmost tree above as

$$t \mapsto t \uparrow ((\uparrow\emptyset) \mapsto t) \mapsto (\uparrow\emptyset) \mapsto t$$

Now define $x \hookrightarrow y = (\uparrow x) \mapsto y$. This allows to express the rightmost tree above as the least upper bound of

$$\{\{t\} \hookrightarrow t, \{\emptyset \hookrightarrow t\} \hookrightarrow \emptyset \hookrightarrow t\}$$

[6] That least upper bound is guaranteed to exist in the 'standard model' of MT

### 4.6 Model building

All prime maps can be expressed using only $t$, $f$, $x \hookrightarrow y$, and the ability to form finite sets of pairwise coherent maps. Furthermore, all maps can be expressed as supprema of prime maps. These properties allow to define e.g.

$$\begin{array}{lcl} t & = & \langle 0 \rangle \\ f & = & \langle 1 \rangle \\ x \hookrightarrow y & = & \langle 2, x, y \rangle \end{array}$$

Then one may define a set $P$ which models prime maps and a set $C$ which models compact maps by defining $P$ and $C$ as the smallest sets such that

1. $P$ contains $t$ and $f$ and also contains $x \hookrightarrow y$ for all $x \in C$ and $y \in P$.
2. $C$ is the set of finite sets of coherent elements of $P$.

In (2) above, one needs to know which elements of $P$ are coherent. The cases where elements of $P$ are coherent are the following:

1. $t \supset t$.
2. $f \supset f$.
3. $f \supset x \hookrightarrow y$ for all $x \in C$ and $y \in P$.
4. $x \hookrightarrow y \supset f$ for all $x \in C$ and $y \in P$.
5. $x \hookrightarrow y \supset x' \hookrightarrow y'$ iff $x \not\supset x' \vee y \supset y'$ for $x, x' \in C$ and $y, y' \in P$.

In (5) above, one needs to know which elements of $C$ are coherent. Two elements $x$ and $y$ of $C$ are coherent when all elements of $x$ are coherent with all elements of $y$.

The rules above allow to define $P$, $C$, and coherence by mutual recursion which allows to construct a model of the M programming language and the axioms and inference rules stated so far.

To get a model of full map theory, one has to use a large cardinal $\kappa$ and define $C$ as the set of sets with cardinality less than $\kappa$ of coherent elements of $P$ (to be precise, one has to use an inaccessible ordinal $\sigma$ and then choose $\kappa$ as a regular ordinal larger than $\sigma$). A consistency proof of map theory based on these ideas can be found in [1].

# 5 Quantification

## 5.1 Choices to be made

Map theory (MT) is an extension of the theory from Section 3. We obtain MT by adding a quantifier to the syntax and by extending the collection of axioms. When doing so, two choices must be made.

First, one has to decide which kind of quantifier to add. The obvious possibilities are the universal quantifier $\forall$, the existential quantifier $\exists$, or Hilberts epsilon operator $\varepsilon$.

Second, one has to decide which domain to quantify over. The quantifier of ZFC quantifies over all sets, so the obvious choice for MT would be to quantify over all maps. That turns out to be a bad idea, however.

## 5.2 Choice of quantifier

Concerning the choice of quantifier, it does not matter whether one chooses $\forall$ or $\exists$ since each one is easy to define from the other. So let us rule out $\exists$.

One can define $\forall$ from $\varepsilon$, but the opposite is not possible, so the choice between $\forall$ and $\varepsilon$ does matter.

Since $\varepsilon$ is stronger than $\forall$, $\varepsilon$ is the most obvious candidate. Including $\varepsilon$ in MT gives the axiom of choice for free. Furthermore, the combination of $\varepsilon$ and the fixed point operator $\mathsf{Y}$ allows to give a particularly easy proof of the well-ordering theorem: one can simply well-order any set by recursive application of $\varepsilon$ in a very natural way. One can say more pro and cons $\varepsilon$, however.

## 5.3 New function letters

The $\varepsilon$ quantifier has a less celebrated property, which may be of increasing importance in the future: It makes it much simpler to introduce new functions in a theory. As an example, in Mendelsons system $S$ (Peano arithmetic, [7]), one can prove the existence and uniqueness of quotient and remainder:

$$\forall a, b \exists_1 q, r\colon b \neq 0 \wedge a = bq + r \wedge r < b \vee b = 0 \wedge q = 0 \wedge r = a$$

According to the meta-theorem of new constants and function letters (c.f. [7]), the theorem above allows to construct an extended system $S'$ with two new function letters, $x/y$ and $x\%y$, and a new axiom,

$$\forall a, b\colon b \neq 0 \wedge a = b(a/b) + a\%b \wedge a\%b < b \vee b = 0 \wedge a/b = 0 \wedge a\%b = a$$

The meta-theorem of new function letters says that if $S$ is consistent then $S'$ is also consistent and is a conservative extension of $S$, so that one may just as well work in $S'$. That works fine for human mathematicians, but it is cumbersome to change theory when doing machine verification. Having $\varepsilon$ one may simply define $x/y$ and $x\%y$ thus:

$$\begin{aligned} a/b &= \varepsilon q \exists r\colon a = bq + r \wedge r < b \\ a\%b &= \varepsilon r \exists q\colon a = bq + r \wedge r < b \end{aligned}$$

These definitions are easy to deal with in mechanical proof systems.

## 5.4 Problems with $\varepsilon$

If one includes $\varepsilon$ in MT then the Axiom of Choice (AC) becomes provable in MT. So objections against AC may be taken as objections against $\varepsilon$.

AC was first formulated by Zermelo even though it had been in use before. Zermelo used AC for proving Zorns lemma, and it is said that Zermelo was taunted for AC since it seemed too easy to prove lemmas if one was allowed to invent new axioms for each new lemma.

The objections against AC seems to be:

- AC is dangerous. AC was formulated during the third fundamental crisis where new dubious axioms were probably not particularly welcome. As ZF and ZFC have been shown to be equiconsistent, however, this objection is no longer valid.
- AC is superflous. At the time of Zermelo, it must have been the hope to be able to prove Zorns lemma without the use of new axioms. As the independence of AC from ZF has been shown, this objection is no longer valid either.
- AC has peculiar consequences like the ability to divide a sphere into two spheres of the same size of the original. This seems to be inevitable for axioms about infinities. Even the axiom of infinity has a peculiar consequence: together with the power set axiom it entails that there exist infinities of different sizes.
- AC is unnecessary. One can develop substantial amounts of mathematics without AC. The same objection can be raised against e.g. the law of excluded middle or the axiom of replacement, and it is of course a matter of taste which assumptions to include in a theory.

It is the purpose of MT to be a convenient foundation for the working mathematician, and for that reason $\varepsilon$ has been included in it. The benefits are that one can make definitions like those in Section 5.3 easily and one can use AC freely. The drawback is that AC is hardwired into MT so that one cannot easily drop it as one can with AC in ZFC.

## 5.5 Strict versus lazy quantifiers

In MT, $\forall$ lives at the level of terms and not at the level of well-formed formulas. In MT, $\forall$ is a function which, when applied to an argument $p$ as in $\forall p$, tests whether or not $px \equiv \mathsf{T}$ for all $x$ in the domain of quantification.

Let $\Phi$ denote the domain of quantification. The domain of quantification is going to be defined in Section 6. Section 6.1 explains why $\Phi$ should not be taken to be the set of all maps. Section 6 chooses a $\Phi$ which is neither the set of all maps, nor the empty set.

Like any other function of MT, $\forall$ must be monotonic such that $p \preceq q$ implies $\forall p \preceq \forall q$. The two most obvious ways of achieving monotonicity are the following *strict* and *lazy* quantifiers:

$$\forall_{\text{strict}} p \quad = \quad \begin{cases} \mathsf{T} & \text{if } px \equiv \mathsf{T} \text{ for all } x \in \Phi \\ \bot & \text{if } px \equiv \bot \text{ for some } x \in \Phi \\ \mathsf{F} & \text{otherwise} \end{cases}$$

$$\forall_{\text{lazy}} p \quad = \quad \begin{cases} \mathsf{T} & \text{if } px \equiv \mathsf{T} \text{ for all } x \in \Phi \\ \mathsf{F} & \text{if } px \not\equiv \mathsf{T}, \bot \text{ for some } x \in \Phi \\ \bot & \text{otherwise} \end{cases}$$

The choice operator $\varepsilon$ is also a function. When applied to an argument as in $\varepsilon p$, it returns an $x \in \Phi$ such that $px \equiv \mathsf{T}$. There has to be two exceptions, however. Firstly, if $px \not\equiv \mathsf{T}$ for all $x \in \Phi$ then $\varepsilon$ has to be excused for not returning an $x \in \Phi$ such that $px \equiv \mathsf{T}$. Secondly, like any other function, $\varepsilon$ has to be monotonic which puts further restrictions on $\varepsilon$. A strict $\varepsilon$ has the following properties:

- $\varepsilon p \equiv \bot$ if $px \equiv \bot$ for some $x \in \Phi$.
- $\varepsilon p \in \Phi$ if $px \not\equiv \bot$ for all $x \in \Phi$.
- $p(\varepsilon p) \equiv \mathsf{T}$ if $px \not\equiv \bot$ for all $x \in \Phi$ and $px \equiv \mathsf{T}$ for some $x \in \Phi$.

A lazy $\varepsilon$ which satisfies

$$p(\varepsilon p) \equiv \mathsf{T} \text{ if } px \equiv \mathsf{T} \text{ for some } x \in \Phi \tag{1}$$

is not tenable. To see that let $a, b \in \Phi$ be compact maps which satisfy $a \not\asymp b$. If (1) holds then

- $a \preceq \varepsilon\chi_a \preceq \varepsilon\lambda x.\mathsf{T}$
- $b \preceq \varepsilon\chi_b \preceq \varepsilon\lambda x.\mathsf{T}$

contradicting $a \not\asymp b$.

It is possible to construct choice operators which are more lazy than the strict one, but they are complicated and difficult to work with. For that reason, we include the strict $\varepsilon$ in MT. That settles the choice of universal quantifier in favor of the strict one so that $\forall$ from now on denotes $\forall_{\text{strict}}$.

## 5.6 The syntax of Map Theory

The syntax of Map Theory is the same as the one defined in Section 2.1 except that $\varepsilon$ is added as a new construct:

$$\begin{array}{lcll} \mathcal{H} & ::= & \varepsilon & \text{(Hilbert epsilon)} \\ \mathcal{T} & ::= & \mathcal{V} \mid \mathcal{N} \mid \mathcal{R} \mid \mathcal{H} & \text{(Term)} \end{array}$$

Section 6 defines the domain $\Phi$ of quantification and also defines a map $\phi$ with the following property:

$$\phi x = \begin{cases} \mathsf{T} & \text{if } x \in \Phi \\ \bot & \text{otherwise} \end{cases}$$

We shall refer to elements of $\Phi$ as *wellfounded maps*. Section 6 defines $\phi$ using the syntax of Map Theory given above. Now define

$$\begin{array}{lcl} !x & \doteq & \mathsf{if}(x, \mathsf{T}, \mathsf{T}) \\ \exists & \doteq & \lambda p.\ \neg\neg(p(\varepsilon p)) \\ \exists x{:}\, P & \doteq & \exists(\lambda x.\ P) \\ \forall x{:}\, P & \doteq & \neg\exists x{:}\, \neg P \\ \forall & \doteq & \lambda p.\ \forall x{:}\, px \\ \varepsilon x{:}\, P & \doteq & \varepsilon(\lambda x.\ P) \end{array}$$

We have that $!x \equiv \mathsf{T}$ iff $x$ is *Boolean*, i.e. iff $x \not\equiv \bot$ and that $\forall p \equiv \mathsf{T}$ if $px \equiv \mathsf{T}$ for all wellfounded $x$. The two negation signs in the definition of $\exists$ ensures that $\exists p$ equals $\mathsf{T}$ or $\mathsf{F}$ or $\bot$. The constructs $\exists x{:}\, P$, $\forall x{:}\, P$, and $\varepsilon x{:}\, P$ are introduced for notational convenience.

### 5.7 Axioms of quantification

The axioms of quantification of Map Theory read:

$$\begin{array}{l} \phi(\varepsilon p) \equiv \forall x{:}\, !(px) \\ !(\forall p) \equiv \forall x{:}\, !(px) \\ \forall p \to \phi x \to px \\ \varepsilon p \equiv \varepsilon x{:}\, \phi x \wedge px \end{array}$$

The first axiom says that $\varepsilon p$ is wellfounded iff $px$ is Boolean for all wellfounded $x$. The second says that $\forall p$ is Boolean under the same condition. The third axiom says that if $py$ is true for all wellfounded $y$ and $x$ is wellfounded then $px$ is true. The last axiom expresses *Ackermanns axiom* ([2], p.244).

The definition of Map Theory is now complete except that the definition of $\phi$ is not yet stated.

## 6 Choice of domain

### 6.1 Size of domain

The choice of domain $\Phi$ remains. To simplify the discussion, we shall consider the properties of $\forall$ instead of $\varepsilon$ for each possible choice of $\Phi$.

If we take $\Phi$ to be the set of all maps then, among other, $\bot \in \Phi$. In this case, monotonicity gives us

$px \equiv \mathsf{T}$ for all $x \in \Phi$ if and only if $p\bot \equiv \mathsf{T}$.

so the quantifier becomes trivial:

$$\forall \equiv \lambda p.\neg\neg(p\bot)$$

Hence, it is reasonable to require $\bot \notin \Phi$. The other extreme would be to let $\Phi$ be the empty set, but then we would have

$$\forall \equiv \lambda p.\mathsf{T}$$

So $\Phi$ should neither be too big, nor too small. To ensure that $\Phi$ is of any use, $\Phi$ must have an infinite subset $\Phi'$ of compact maps such that for all $x, y \in \Phi', x \neq y$ we have

- $x \not\subset y$
- $(x \downarrow y) \notin \Phi$

## 6.2 Representation of sets

When MT was constructed first time [3], it was easy to see that $\Phi$ should not contain maps like $\bot$ and $\lambda x.\bot$ but could safely contain maps like $\mathsf{T}$ and $\lambda x.\lambda y.\mathsf{T}$.

One of the intensions of MT was to build a foundation on top of a programming language which had the same strength as ZFC. For that reason, $\Phi$ had to contain a subset $\Phi'$ with the properties stated in Section 6.1 with the further property that every set in the universe of ZFC should be representable by an element of $\Phi'$. Furthermore, it should be possible to model the membership relation of ZFC by a function in MT.

To achieve this it is reasonable to consider how to represent sets of ZFC by maps of MT. An obvious choice would be to let a map $m$ represent the set $\{x|mx \equiv \mathsf{T}\}$. Trying that was unfruitful, however.

A representation of sets which turned out to work was the following: let $\mathsf{T}$ represent the empty set. Furthermore, if $m \in \Phi$ and $m \neq \mathsf{T}$ then let $m$ represent the set $\{mx|x \in \Phi\}$. This representation happens to work, and it immediately hints at the following two properties for $\Phi$:

- $\mathsf{T} \in \Phi$.
- $x, y \in \Phi \Rightarrow xy \in \Phi$.

So $\Phi$ should contain $\mathsf{T}$ and be closed under application.

## 6.3 Wellfoundedness

The axiom of *restriction* or *wellfoundedness* of ZFC hints at another property, namely that elements of $\Phi$ should be *wellfounded* in a certain sense.

We shall say that a map $g$ is wellfounded w.r.t. a set $G$ of maps if, for all infinite sequences $x_0, x_1, x_2, \cdots \in G$ there exists a natural number $n$ such that

$$gx_0x_1\cdots x_{n-1} \equiv \mathsf{T}$$

For all sets $G$ of maps let $G^\circ$ denote the set of maps that are wellfounded w.r.t. $G$. The chosen representation of sets together with the axiom of wellfoundedness entail that all elements of $\Phi$ should be wellfounded w.r.t. $\Phi$:

$$\Phi \subseteq \Phi^\circ$$

It would be convenient to have $\Phi = \Phi^\circ$. To see that $\Phi = \Phi^\circ$ is not tenable, however, we prove $\mathsf{I} \in \Phi^\circ$ and $\mathsf{I} \notin \Phi$ where $\mathsf{I} = \lambda x.\ x$.

$\mathsf{I} \in \Phi^\circ$: Suppose $x_0, x_1, x_2, \cdots \in \Phi$. We shall find a natural number $n$ such that $\mathsf{I} x_0 x_1 \cdots x_n \equiv \mathsf{T}$. From $x_0 \in \Phi \subseteq \Phi^\circ$ we have $x_0 \in \Phi^\circ$. Hence, there exists a natural number $n$ such that $x_0 x_1 x_2 \cdots x_n \equiv \mathsf{T}$ so $\mathsf{I} x_0 x_1 \cdots x_n \equiv \mathsf{T}$.

$\mathsf{I} \notin \Phi$: Suppose $\mathsf{I} \in \Phi$. We have $\mathsf{I}, \mathsf{I}, \mathsf{I}, \ldots \in \Phi$ so $\mathsf{I} \in \Phi^\circ$ entails that there exists a natural number $n$ such that

$$\mathsf{I}\overbrace{\mathsf{I}\mathsf{I}\cdots\mathsf{I}}^{n} \equiv \mathsf{T}$$

But the left hand side of the equation above equals $\mathsf{I}$ regardless of the value of $n$ which gives a contradiction. Hence, $\mathsf{I} \notin \Phi$. This together with $\mathsf{I} \in \Phi^\circ$ gives

$$\Phi \subset \Phi^\circ$$

## 6.4 Continuity

From $\mathsf{I} \in \Phi^\circ$ and $\mathsf{I} \notin \Phi$ it is obvious that $\Phi = \Phi^\circ$ is not tenable, but there is another reason why one should expect $\Phi = \Phi^\circ$ to have no solutions: one would expect the cardinality of $\Phi^\circ$ to be equal to that of the powerset of $\Phi$ and, hence, greater than that of $\Phi$. Nevertheless, if $M$ denotes the set of all maps, then the following properties would be convenient:

$$\begin{array}{lrcl} (6.4.1) & \Phi & \subseteq & M \\ (6.4.2) & \Phi & = & \Phi^\circ \\ (6.4.3) & M & = & (M \to M) \cup \{\mathsf{T}, \bot\} \end{array}$$

Equation (6.4.3) is not tenable because the set $M \to M$ of functions from $M$ to $M$ has greater cardinality than $M$. So both (6.4.2) and (6.4.3) are impossible for cardinality reasons.

The usual countermeasure for a domain equation like (6.4.3) is to restrict $M \to M$ to functions which are continuous in some sense. For ordinary models of lambda-calculus and related theories it is customary to use Scott-continuity. For models of map theory one has to go to the generalized notion of $\kappa$-Scott-continuity where $\kappa$ is an ordinal.

So one may try the same on (6.4.2) and restrict $\Phi^\circ$ to continuous functions. But the notion of continuity used in (6.4.2) must be different from that used in (6.4.3) because the identity function $\mathsf{I}$ must be continuous in the sense used in (6.4.3) and discontinuous in the sense used in (6.4.2).

## 6.5 $\kappa$-Scott Continuity

The simplest way to explain $\kappa$-Scott continuity is as follows: We say that $c$ is a $\kappa$-chain if $c \in \kappa \to M$ and $\forall\beta{\in}\kappa\forall\alpha{\in}\beta\colon c(\alpha) \preceq c(\beta)$. We require $M$ to be $\kappa$-complete in the sense that every $\kappa$-chain $c$ has a supremum $\uparrow c$ in $M$. Then a function $g \in M \to M$ is $\kappa$-continuous if

$$g(\uparrow c) = \uparrow(g \circ c)$$

for all $\kappa$-chains $c$. This is equivalent to ordinary Scott-continuity for $\kappa = \omega$.[7]

If two ordinals $\kappa$ and $\kappa'$ have the same co-finality then the notions of $\kappa$- and $\kappa'$-continuity are identical. Hence, it is reasonable to restrict $\kappa$ to be regular, i.e. restrict $\kappa$ to be equal to its own co-finality.

Now let $A \xrightarrow{\kappa} B$ denote the set of $\kappa$-Scott continuous functions from $A$ to $B$. If $\kappa$ increases, then the notion of $\kappa$-Scott continuity becomes weaker and, hence, $A \xrightarrow{\kappa} B$ becomes bigger. Hence, to get a big universe for map theory one should select a large cardinal $\kappa$ and replace (6.4.3) by the following:

$$(6.4.3') \quad M \quad = \quad (M \xrightarrow{\kappa} M) \cup \{\mathsf{T}, \bot\}$$

## 6.6 Uniform continuity

For all sets $S$ let $S^*$ denote the set of finite lists of elements of $S$. For $m \in M$ and $x = \langle x_1, \ldots, x_n\rangle \in M^*$ let $mx$ denote $mx_1 \cdots x_n$. For all sets $U$ and $V$ we shall say that $U$ is *$V$-small* if the cardinality of $U$ is less than the cardinality of $V$. Let $\mathcal{P}_V(S)$ denote the set of $V$-small subsets of $S$. For all $S \subseteq M$ define $\Uparrow S = \{y \in M | \exists x \in S\colon x \preceq y\}$. For all $x \in M$ and $S \subseteq M$ define the neighborhood

$$B(x, S) = \{y{\in}M | \forall s{\in}S^*\colon \neg\neg(xs) \preceq \neg\neg(ys)\}.$$

The set $\{B(x, S) | x \in M \wedge S \in \mathcal{P}_\kappa(M)\}$ of neighborhoods generates the $\kappa$-Scott topology. A function $g \in M \to M$ is $\kappa$-continuous iff

$$\forall x{\in}M\forall U{\in}\mathcal{P}_\kappa(M)\exists V{\in}\mathcal{P}_\kappa(M)\forall y{\in}B(x, V)\colon g(y) \in B(g(x), U)$$

We now define that a function $g \in M \to M$ is *uniformly* $\kappa$-continuous iff

$$\forall U{\in}\mathcal{P}_\kappa(M)\exists V{\in}\mathcal{P}_\kappa(M)\forall x{\in}M\forall y{\in}B(x, V)\colon g(y) \in B(g(x), U)$$

Let $S^\circ_\kappa$ denote the set of maps which are wellfounded w.r.t. $S$ and are uniformly $\kappa$-continuous. We are now in a position to reformulate (6.4.2): Let $\sigma$ be an inaccessible ordinal, let $\kappa$ be a regular cardinal greater than $\sigma$. Instead of (6.4.2) we shall require:

$$(6.4.2') \quad \Phi \quad = \quad \Uparrow \Phi^\circ_\sigma$$

[7] $\kappa$-Scott-domains are $\kappa$-algebraic and continuity may be defined via commutation with suprema of $\kappa$-chains as is done e.g. in [6] for $\omega$-continuity, in [8] for $\omega_1$-continuity, and in [1] for $\kappa$-continuity.

## 6.7 A property of $\Phi$

The treatment given so far hints at a way to construct a model for Map Theory and hints at a way to define $\Phi$. The model is developed in [1], but the definition of $\Phi$ given there is a bit more convenient to work with: $\Phi$ is taken to be the smallest subset of $M$ for which

$$G \in \mathcal{P}_\sigma(\Phi) \Rightarrow (G^\circ \to G) \subseteq \Phi$$

The definition of $\Phi$ above is the one we shall formalize in the following.

## 6.8 Classes

In the following, we use maps $m$ to represent classes $\{x|mx \equiv \mathsf{T}\}$. As mentioned in Section 6.2, sets of ZFC are represented another way, and we use the term *class* in the following to distinguish from ZFC sets. The classes considered in the following are not the same as those of NBG set theory. Now define:

| | | | |
|---|---|---|---|
| $x \sqsubset y$ | $\doteq$ | $yx$ | (class membership) |
| $\{x\|P\}$ | $\doteq$ | $\lambda x.P$ | (class comprehension) |
| $V$ | $\doteq$ | $\{x\|\mathsf{T}\}$ | (universal class) |
| $\{\mathsf{T}\}$ | $\doteq$ | $\{x\|x\}$ | (class containing only $\mathsf{T}$) |
| $\bigcup x \sqsubset G\colon H$ | $\doteq$ | $\{m\|\mathsf{E}x\colon x \sqsubset G \wedge m \sqsubset H\}$ | (Union class) |

.

In $\{x|P\}$ and $\bigcup x \sqsubset G\colon H$ above, $x$ may occur free in $P$ and $H$.

## 6.9 The dual class

In line with Section 6.3, let $G^\circ$ denotes the class of maps which are wellfounded w.r.t. the class G. We have $G \preceq H \Rightarrow H^\circ \preceq G^\circ$ so monotonicity prevents us from defining $G^\circ$ in map theory. But we can define a universal quantifier which quantifies over $G^\circ$ and a set $G^\circ \to H$ of functions with domain $G^\circ$ and range $H$. To do so we define $g|G$ such that $g|G$ is the function $g$ recursively restricted to the class $G$.

| | | |
|---|---|---|
| $g\|G$ | $\doteq$ | $\mathsf{if}(g, \mathsf{T}, \lambda x.\ \mathsf{if}(x \sqsubset G, (gx)\|G, \bot))$ |
| $\forall x \sqsubset G^\circ\colon P$ | $\doteq$ | $\forall x\colon (\lambda x.P)(x\|G)$ |
| $G^\circ \to H$ | $\doteq$ | $\{g\|\forall x \sqsubset G^\circ\colon gx \sqsubset H\}$ |

Whenever $g \in \Phi$ and $G \subseteq \Phi$ we have $g|G \in G^\circ$. The definition of $\forall x \sqsubset G^\circ\colon P$ references $\forall$ which already quantifies over wellfounded maps, so $\forall x \sqsubset G^\circ\colon P$ quantifies over $G^\circ$ whenever $G \subseteq \Phi$. $G^\circ \to H$ is the set of functions with domain $G^\circ$ and range $H$ whenever $G \subseteq \Phi$. We may also define a lazy and a strict union:

| | | |
|---|---|---|
| $\biguplus x \sqsubset G^\circ\colon H$ | $\doteq$ | $\{m\|\mathsf{E}x\colon \phi x \wedge m \sqsubset (\lambda x.H)(x\|G)\}$ |
| $\bigcup x \sqsubset G^\circ\colon H$ | $\doteq$ | $(\forall x \sqsubset G^\circ\colon !H) : (\biguplus x \sqsubset G^\circ\colon H)$ |

The $\uplus$ operator is a lazy union which treats $\bot$ as the empty set. The $\cup$ operator is a more strict union which treats $\bot$ as undefined but treats $\lambda x.\bot$ as the empty set.

### 6.10 The definition of $\phi$

To define $\phi$ we first define $C_a$ and $G_a$. $C_a$ and $G_a$ are defined such that $C_a, G_a \in \mathcal{P}_\sigma(\Phi)$ for all maps $a$.

$$\begin{array}{lcl} C_a & \doteq & G_a^\circ \to G_a \\ G_a & \doteq & \mathsf{if}(a, \{\mathsf{T}\}, \bigcup x \sqsubset C_{a\mathsf{T}}^\circ \colon C_{a\mathsf{F}x}) \\ \phi & \doteq & \bigcup a \sqsubset V \colon C_a \end{array}$$

An inaccessible ordinal $\sigma$ has the property that if a set $S$ is $\sigma$-small then the powerset $\mathcal{P}(S)$ is also $\sigma$-small. That ensures that if $G_a$ is $\sigma$-small then so is $C_a$. Furthermore, the union of a $\sigma$-small set of $\sigma$-small sets is in turn $\sigma$-small. That ensures that if $C_{a\mathsf{T}}$ is $\sigma$-small and if $C_{a\mathsf{F}x}$ is $\sigma$-small for all $x \in C_{a\mathsf{T}}^\circ$ then $G_a$ is $\sigma$-small. The strict union operator $\cup$ is used in the definition of $G_a$ to avoid problems with maps $a$ for which $a\mathsf{F}b \equiv a$ for some $b \in C_{a\mathsf{T}}^\circ$.

# 7 Conclusion

The axioms and inference rules of Map Theory have been introduced. The axiomatic system is considerably simpler than that of [3]. The model of Map Theory given in [1] is supposed to carry over to the formulation given here. The development of ZFC given in [5] is also supposed to carry over to the formulation given here. The last two claims, however, are work of the future.

# Index


?, 8
$G^\circ \to H$, 23
$\bot$, 8
$\{\mathsf{T}\}$, 23
$x \propto \langle y|v{:=}z\rangle$, 8
$\lambda x.y$, 4
$\neg x$, 9
$\{x|P\}$, 23
$g|G$, 23
$x \circ y$, 8
$x \to y$, 8
$x \equiv y$, 8
$x \downarrow y$, 10
$x \mapsto y$, 14
$x \| y$, 4
$x \preceq y$, 10
$x \sqsubset y$, 23
$x \wedge y$, 9
$G^\circ$, 21
$S^*$, 22
$\Uparrow S$, 22
$!x$, 19
$\uparrow S$, 14
$x : y$, 14
$x \approx y$, 7
$x \backsim y$, 14
$x \uparrow y$, 14
$x \hookrightarrow y$, 14

$\forall x \sqsubset G^\circ : P$, 23
$\forall$, 16, 19
Ackermanns axiom, 19
application, functional, 4
axiom, Ackermanns, 19

Boolean, 19
branch, 4

$\mathcal{C}$, 4
$C_a$, 24
class, 23
coherent maps, 14
compact, 12
continuity, $\kappa$-Scott, 22
continuity, uniform, 22

disjunction, parallel, 4

$\mathsf{E}x$, 4
$\varepsilon$, 16
$\exists$, 16, 19
existence, pure, 4
extensionality, 11

f, 12
$\mathsf{F}$, 9
$\Phi$, 17
$\mathcal{F}$, 4
$\phi$, 19, 24
$f_*$, 5
$f$, 14
$f_1$, 5
$f_{\mathcal{A}}$, 6
$f_{\mathcal{B}}$, 6
$f_{\mathcal{D}}$, 6
$f_{\mathcal{E}}$, 7
function, 4
function, partial, 5
function, total, 5
functional application, 4

$G_a$, 24

$\mathsf{I}$, 11
$\mathsf{if}(x, y, z)$, 4

lazy quantifier, 18
$\mathrm{lub}(x, y, z)$, 14

M, 4
map, 8
map, prime, 13
map, wellfounded, 19
maps, coherent, 14
MT, 2

normal form, 4

object variable, 4

$\mathcal{P}_\alpha(S)$, 22
parallel disjunction, 4
partial function, 5
prime map, 13
pure existence, 4

QND, 9
quantifier, lazy, 18
quantifier, strict, 18
Quartum Non Datur, 9

reduce, 5
reduction rule, 5
reduction sequence, 5
reduction step, 5
rule, reduction, 5

$\kappa$-Scott continuity, 22
sequence, reduction, 5
$S$-small, 22
step, reduction, 5
strict quantifier, 18

t, 12
$\mathsf{T}$, 4
$\mathcal{T}$, 4
$t$, 14
term, 4
total function, 5

$\bigcup x{\sqsubset}G\colon H$, 23
$\bigcup x{\sqsubset}G^\circ\colon H$, 23
$\biguplus x{\sqsubset}G^\circ\colon H$, 23
$\mathcal{U}$, 4
uhr element, 4
uniform continuity, 22

$V$, 23
$\mathcal{V}$, 4
variable, object, 4

wellfounded, 20
wellfounded map, 19

$\mathsf{Y}$, 10

ZFC, 2

# References


[1] C. Berline and K. Grue. A $\kappa$-denotational semantics for Map Theory in ZFC+SI. *Theoretical Computer Science*, 179(1–2):137–202, June 1997.

[2] U. Felgner. Choice functions on sets and classes. In *Sets and Classes: On the works by Paul Bernays*, pages 217–255. North-Holland, 1976.

[3] K. Grue. Map theory. *Theoretical Computer Science*, 102(1):1–133, July 1992.

[4] K. Grue. Lambda-calculus as a foundation for mathematics. In C. Anthony Anderson and Michael Zeleny, editors, *Logic, Meaning and Computation : Essays in Memory of Alonzo Church*, volume 305 of *Synthese Library*, pages 289–314, Dordrecht, 2002. Kluwer Academic Publishers.

[5] K. Grue. Map theory with classical maps. Technical Report 02/21, DIKU, Universitetsparken 1, DK-2100 Copenhagen, Denmark, 2002.

[6] J.L. Krivine. *Lambda-calculus, types and models.* Ellis & Horwood, 1993.

[7] E. Mendelson. *Introduction to Mathematical Logic.* Wadsworth and Brooks, 3. edition, 1987.

[8] G. Plotkin. A power domain for countable non-determinism. In *Lecture Notes in Computer Science*, volume 140, pages 418–428. ICALP'82, Springer-Verlag, 1982.