Recursive Data Types in Typed Set Theory

R.D. Arthan
Lemma 1 Ltd.
rda@lemma-one.com

Abstract

Appeals to the axiom of replacement in classical set theory cannot automatically be carried over into simple type theory or other typed languages such as the Z specification language [6]. If, like Z, a typed language provides a mechanism for defining sets by recursion equations, then many of the basic results in the literature for the untyped case do not carry over directly. This paper considers the consistency of such definitions in a class of typed systems broad enough to include Z. In an earlier paper, commonly used criteria for consistency in the untyped case were shown to be sufficient for consistency relative to the Zermelo axioms with AC; here we improve upon the earlier work by eliminating the appeals to AC.

1 INTRODUCTION

Logical languages intended for use in computer systems specification commonly provide mechanisms for defining classes of recursive data structures. For example, in the Z specification language, we can give the following definition of a class of (balanced) trees.

$$\mathsf{BALTREE} ::= \mathsf{Leaf} \ll \mathbb{N} \gg | \ \mathsf{Node} \ll \{b_1, b_2 : \mathsf{BALTREE} | b_1 = b_2\} \gg \tag{1}$$

This introduces a new recursively defined set, $\mathsf{BALTREE}$, together with its constructor functions $\mathsf{Leaf} : \mathbb{N} \rightarrowtail \mathsf{BALTREE}$ and $\mathsf{Node} : \{b_1, b_2 : \mathsf{BALTREE} | b_1 = b_2\} \rightarrowtail \mathsf{BALTREE}$. The constructor functions are intended to display the set $\mathsf{BALTREE}$ as the least fixed point of the following recursion equation in the language of first-order set theory:

$$X = \omega \sqcup \{(b_1, b_2) \in X \times X | b_1 = b_2\} \tag{2}$$

The generality required in this type of definition means that there is no simple syntactic criterion for the consistency of the definition. Indeed, if the (closed) formula $\mathcal{P}$ is undecidable, then we will be unable to prove or disprove the consistency of a definition such as:

$$\mathsf{UNDEC} ::= \mathsf{UnDec} \ll \{A : \mathbb{P}\mathsf{UNDEC} | \mathcal{P}\} \gg \tag{3}$$

In the light of such examples, the best one can hope for is for convenient sufficent conditions for consistency that work for the cases that tend to occur in practice. The main purpose of this paper is to justify the most commonly-used sufficient conditions for the existence of solutions of recursion equations such as (2). The standard mathematical justifications require the axiom of replacement and so are at odds with the type system of languages such as Z. In an earlier paper, the use of replacement was eliminated by recourse to the axiom of choice [1]. In the present paper, we carry out the necessary constructions without using choice — and so restore some of the structural insights offered by the standard proofs in the untyped case using replacement.

Most operators that arise in practice are finitary, and our results are as just advertised for the finitary case. However, much of the theory and some of the early literature for Z works with operators satisfying the weaker condition of continuity. For a general continuous operator, we find that we still need to appeal to the axiom of dependent choices (DC) or something like it. However, we identify a useful class of operators (the monotonic operators having finite content) lying somewhere between the continuous operators and the finitary operators for which no use of choice or dependent choices is needed. In any case, we know of no common use in systems specification practice of Z free type definitions that are not finitary.

The structure of the paper is as follows:

- Section 2 gives a very brief summary of the standard theory in ZF of inductive definitions and introduces terminology for later use, in particular the notions of finitary and continuous operators.
- Section 3 defines our notion of a typable operator.
- Section 4 defines, for each type, a maximal typable operator of that type. The notion of an operator having finite content is introduced and it is shown that monotonic operators with finite content are also subsumed by the maximal typable operators as also are finitary typable operators. The same holds for continuous operators provided we admit the axiom of dependent choices. This reduces the general consistency problem to a single specific problem in each type.
- Section 5 proves that the maximal typable operators possess fixed points from which we conclude that a class of operators including the finitary operators and many non-finitary but continuous operators have fixed points.
- Section 6 gives some concluding remarks.

While this paper is intended to be self-contained, we refer the reader to [1] for a more leisurely exposition of the theoretical framework and its import for Z.

2 Preliminaries

Pausing only to note that we will conform to the standard use of the term "finitary" rather than that used in [5] and that we now eschew the pun whereby continuous operators were called "cts" in [1], let us fix some terminology for the rest of the paper:

- an *operator* is a term, ϕ, or, for emphasis, $\phi(X)$, in the language of first order set theory in which X may appear free[1];
- an operator ϕ is *monotonic* if it satisfies the condition[2]:

$$\forall X \bullet \forall Y \bullet Y \subseteq X \implies \phi(Y) \subseteq \phi(X); \tag{4}$$

- if ϕ is an operator, a set X is *ϕ-closed* if $\phi(A) \subseteq X$ for every $A \subseteq X$.
- an operator ϕ is *continuous* if it commutes with countable unions of chains, i.e. for any chain $X_1 \subseteq X_2 \subseteq \ldots$, we have:

$$\phi(\bigcup_i X_i) = \bigcup_i \phi(X_i); \tag{5}$$

- finally, an operator ϕ is *finitary*, if it satisfies the condition:

$$\forall X \bullet \phi(X) = \bigcup\{\phi(Y) | Y \subseteq X \land Y \text{ is finite}\}. \tag{6}$$

Finitary operators are easily seen to be continuous. Continuous operators may be shown to be monotonic (remembering that $A \subseteq B$ iff. $B = A \cup B$ and considering the chain $A \subseteq B \subseteq B \subseteq B \ldots$).

The central problem of this paper is the following: given an operator ϕ, under what circumstances is there a set A which acts as a least fixed point of the recursion equation $X \simeq \phi(X)$. I.e. A must enjoy the following properties:

- there is a bijection f mapping $\phi(A)$ to A.
- for no proper subset B of A does the restriction of f to $\phi(B)$ map $\phi(B)$ into B.

If these conditions hold, we say A is a least fixed point for ϕ up to a bijection and write $A \simeq_{\mathsf{LFP}} \phi(A)$.

Now in the usual theory of monotonic operators in ZF one is concerned with fixed points "on the nose" rather than up to a bijection (i.e., f above must be the identity function). However, this is no great restriction, since there are standard constructions for delivering a fixed point for a monotonic operator that has one and these constructions deliver fixed points on the nose. The construction we will focus on in this paper is the construction "from below" and goes as follows in ZF.

Lemma 1 (ZF) *Given a monotonic operator, ϕ, define a sequence of sets, T_α by transfinite induction as follows:*

$$\begin{aligned} T_0 &= \emptyset & (7)\\ T_{\alpha+1} &= \phi(T_\alpha) & (8)\\ T_\lambda &= \bigcup_{\alpha<\lambda} T_\alpha & (9) \end{aligned}$$

[1]For convenience, we will assume that the first-order language includes all the usual operators of set theory: $\cup$, $\mathbb{P}$ etc. We will use the Z-like notation $\mathsf{seq}\, A$ to denote the set of finite sequences on a set A, represented concretely as finite functions from an interval $1 \mathinner{..} n$ to A (with n ranging over natural numbers). If s is such a sequence, and $i \in \mathsf{dom}(s)$, we will write s_i for the i-th element of s. The idiom $\mathsf{ran}(s)$ gives the set of members of the sequence s.

[2]Here and throughout, we adopt the convention that quantifiers have low precedence: $\forall X \bullet A \implies B$ is a universal quantification not an implication.

where α (resp. λ) ranges over ordinals (resp. limit ordinals). If for some γ, we have $T_\alpha = T_\gamma$ for all $\alpha \geq \gamma$, then T_γ is a least fixed point (on the nose) for ϕ.

Proof: We have $\phi(T_\gamma) = T_{\gamma+1} = T_\gamma$, so T_γ is a fixed point. By transfinite induction on α it is not hard to see that the T_α form a (transfinite) chain, i.e., $T_\alpha \subseteq T_\beta$ whenever $\alpha \leq \beta$. By another induction, we can show that $T_\gamma \subseteq X$ whenever $\phi(X) = X$, so T_γ is a least fixed point. $\square$

For example, if ϕ is continuous, it is not hard to see that the chain must stabilise at or before the first infinite ordinal, ω, so the countable union $T_0 \cup T_1 \cup \ldots$ will provide a fixed point.

Now, we are concerned here with languages, like Z, which impose a type discipline (with say, cartesian product and power set as the type constructors). The natural models for these languages lie within the set $V_{\omega+\omega}$ that gives the minimal model for Zermelo set theory (i.e., ZF without replacement). Unfortunately, the above construction makes essential use of the axiom of replacement to form the unions needed to define the sets T_λ. Moreover, the construction can easily take us outside $V_{\omega+\omega}$ even when the operator ϕ does have a fixed point up to a bijection (see [1] for an example). Thus we can't hope for an adequate supply of fixed points on the nose. From now on we will only be concerned with fixed points up to a bijection, and so in the sequel, when we say "fixed point" that is what we shall mean.

It turns out that as soon as we have found some set X for which $\phi(X)$ is equinumerous with a subset of X, then we can find a least fixed point within X. We record this in the following lemma.

Lemma 2 *Let ϕ be a monotonic operator and X a set such that there is an injection $f : \phi(X) \rightarrowtail X$, then for some $Y \subseteq X$, $\phi(Y) \simeq_{\mathsf{LFP}} Y$.*

Proof: The proof is a fairly straightforward exercise in using the definitions, see [1] for details. $\square$

As a corollary of the lemma, we observe that, if ϕ and ψ are monotonic operators for which $\psi(X) \subseteq \phi(X)$ for all X and ϕ has a fixed point, then so does ψ. Indeed, if T is a fixed point for ϕ, then as $\phi(T) \subseteq \psi(T)$, T will serve for X in the lemma.

3 Typable Operators

We are interested in the operators that can arise in languages obeying a type discipline. The type-forming operators we allow are cartesian product, disjoint union and power set. More precisely, we work with the following system of types:

1. The symbols X and ω are types
2. If τ_1 and τ_2 are types then so are $(\tau_1 \times \tau_2)$, $(\tau_1 \sqcup \tau_2)$, and $(\mathbb{P}\tau_1)$.
3. Nothing is a type except by virtue of rule 1 or rule 2.

Given a set Y and a type τ, we define the semantic value of the type τ at Y, which we write as $\tau(Y)$, using the following rules:

$$
\begin{aligned}
X(Y) &= Y && (10)\\
\omega(Y) &= \omega && (11)\\
(\tau_1 \times \tau_2)(Y) &= \tau_1(Y) \times \tau_2(Y) && (12)\\
(\tau_1 \sqcup \tau_2)(Y) &= \tau_1(Y) \sqcup \tau_2(Y) && (13)\\
(\mathbb{P}\tau)(Y) &= \mathbb{P}(\tau(Y)) && (14)
\end{aligned}
$$

where ω is the set of natural numbers, $\mathbb{P}$ is the usual power set operator, $\times$ is binary cartesian product, and $\sqcup$ is binary disjoint union. We assume here that some fixed functorial constructions for products and disjoint unions have been chosen — the usual ones will do nicely (with $A \sqcup B$ a subset of $(A \cup B) \times \{0,1\}$). The product and disjoint unions must also be constructible in Zermelo set theory (i.e. without the axiom of replacement) — again the usual constructions are adequate.

For any type τ, $Y \mapsto \tau(Y)$[3] is the objects part of a functor from Set to Set. The morphisms part of the functor is given by $(f : X \to Y) \mapsto (\hat{\tau}(f) : \tau(X) \to \tau(Y))$ where $\hat{\tau}(f)$ is defined by the following rules:

$$
\begin{aligned}
\hat{X}(f) &= f && (15)\\
\hat{\omega}(f) &= n \mapsto n && (16)\\
(\tau_1 \hat{\times} \tau_2)(f) &= (x_1, x_2) \mapsto ((\hat{\tau_1}(f))(x_1), (\hat{\tau_2}(f))(x_2)) && (17)\\
(\tau_1 \hat{\sqcup} \tau_2)(f) &= \iota_j(x) \mapsto (\hat{\tau_j}(f))(x) && (18)\\
(\hat{\mathbb{P}\tau})(f) &= s \mapsto \{y : \tau(Y) | \exists x \in s \bullet (\hat{\tau}(f))(x) = y\} && (19)
\end{aligned}
$$

where the ι_j $(j = 1, 2)$ are the injections of the summands into the disjoint union. Intuitively, given a set A and $t \in \tau(A)$, the syntactic structure of τ allows us to think of t as a possibly infinite tree with leaves labelled with elements of A. Since we will need another notion of tree in the sequel, we will use the term τ-tree for this view of an element of $\tau(A)$. If f is a function from A to some set B, then $\hat{\tau}(f)$ is the function which sends $t \in \tau(A)$ to the element of $\tau(B)$ obtained by changing each leaf label, x in the τ-tree t to $f(x)$.

By an easy induction over the structure of τ, for any τ, $X \mapsto \tau(X)$ is monotonic and $(f : X \to Y) \mapsto (\hat{\tau}(f) : \tau(X) \to \tau(Y)$ preserves injections, surjections and bijections.

We now define an operator, $\phi(X)$, to be *typable* with type τ iff. we can prove $\forall X \bullet \phi(X) \subseteq \tau(X)$. Here are two examples of typable operators, both with type $(\omega \sqcup (X \times X))$.

$$
\begin{aligned}
\phi_1(X) &= \omega \sqcup \{(b_1, b_2) \in X \times X | b_1 = b_2\} && (20)\\
\phi_2(X) &= \omega \sqcup \{(b_1, b_2) \in X \times X | b_1 \neq b_2\} && (21)
\end{aligned}
$$

Here ϕ_1 corresponds to the example in section 1. Note that ϕ_1 is a subfunctor of τ, i.e., for any $f : X \to Y$, the restriction, $\hat{\phi_1}$ say, of $\hat{\tau}$ to $\phi_1(X)$ maps $\phi_1(X)$ to $\phi_1(Y)$, thus ϕ_1 and $\hat{\phi_1}$ give the objects and morphisms parts of a functor from Set to Set. This is not the case with

[3]Readers familiar with the Z notation should be warned that the symbol $\mapsto$ is used in this paper with its usual mathematical value: $x \mapsto y$ means $\lambda x \bullet y$, not (x, y).

ϕ_2; for example, if f is the only possible function from the two point set $\mathbf{2}$ to the one point set $\mathbf{1}$, and x is the element $\iota_2(0,1)$ of $\phi(\mathbf{2})$, then $\hat{\tau(f)}(x) = \iota_2(0,0)$ which is not in $\phi_2(\mathbf{1})$ (indeed, the right-hand summand in $\phi(\mathbf{1})$ is empty).

4 Maximal Typable Operators

We now consider typable continuous operators. We will show that for each type, τ, there is a continuous, indeed finitary, operator, ϕ_τ, of type τ, such that for any finitary operator, ϕ, of type τ, and any set X, $\phi(X) \subseteq \phi_\tau(X)$. If we admit the axiom of dependent choices then the same will be true under the weaker assumption that ϕ is continuous. We will also introduce the notion of an operator having finite content. This notion is intermediate between finitariness and continuity and our main results hold for monotonic typable operators having finite content without using dependent choices.

It will be convenient to have a function that determines, for an element t of $\tau(X)$, the raw material in X out of which t is constructed. For each type τ and each set Y, the following equations define by induction over the structure of τ a function $\mathsf{content}_\tau^Y$ to do just that.

$$
\begin{aligned}
\mathsf{content}_X^Y &= x \mapsto \{x\} && (22)\\
\mathsf{content}_\omega^Y &= n \mapsto \emptyset && (23)\\
\mathsf{content}_{(\tau_1 \sqcup \tau_2)}^Y &= \iota_j(x) \mapsto \mathsf{content}_{\tau_j}^Y(x) && (24)\\
\mathsf{content}_{(\tau_1 \times \tau_2)}^Y &= (x_1, x_2) \mapsto \mathsf{content}_{\tau_1}^Y(x_1) \cup \mathsf{content}_{\tau_2}^Y(x_2) && (25)\\
\mathsf{content}_{(\mathbb{P}\tau)}^Y &= s \mapsto \bigcup\{\mathsf{content}_\tau^Y(x) | x \in s\} && (26)
\end{aligned}
$$

where as usual the ι_j $(j = 1, 2)$ are the injections of the summands into the disjoint union.

We now note that the value of $\mathsf{content}_\tau^Y(t)$ is independent of the Y we choose (provided we choose it so that $t \in \tau(Y)$), i.e. $\mathsf{content}_\tau^Y(x)$ is monotonic in Y.

Lemma 3 *If $A \in \tau(X)$ and $C = \mathsf{content}_\tau^X(A)$, then $A \in \tau(C)$.*

Proof: The proof is straightforward by induction over the structure of τ. See [1] for more detail. $\square$

The following lemma provides an alternative characterisation of $\mathsf{content}_\tau^Y$:

Lemma 4 *The following equation holds for any set Y, type τ and element t of $\tau(Y)$:*

$$\mathsf{content}_\tau^Y(t) = \bigcap\{A | A \subseteq Y \wedge t \in \tau(A)\} \qquad (27)$$

Proof: Again this follows from a routine induction over the structure of τ. Use lemma 3 in the inductive steps for cartesian product and power set. $\square$

For any type, τ, the operator ϕ_τ sends a set X to the set of all elements of $\tau(X)$ that only involve a finite part of X. I.e., we define:

$$\phi_\tau(X) = \{A \in \tau(X) | \mathsf{content}_\tau^X(A) \text{ is finite}\} \qquad (28)$$

So, for example:

$$\begin{aligned} \phi_\omega(X) &= \omega && (29)\\ \phi_X(X) &= X && (30)\\ \phi_{(\mathbb{P}X)}(X) &= \mathbb{F}(X) && (31)\\ \phi_{(\mathbb{P}(X\times\omega))}(X) &= \{R : \mathbb{P}(X \times \omega))|\mathsf{dom}(R)\text{ is finite}\} && (32) \end{aligned}$$

The following alternative characterisation of ϕ_τ follows immediately from lemma 4:

Lemma 5 *The following equation holds for any type τ:*

$$\phi_\tau(X) = \bigcup\{\tau(A)|A \subseteq X \wedge A \textit{ is finite}\} \tag{33}$$

□

Given lemma 3, we can conveniently drop the superscript from $\mathsf{content}^Y_\tau(x)$ in cases where an appropriate value of Y can be inferred from the context. Using this convention, we way that a typable operator ϕ of type τ *has finite content* iff. for all sets A, $\mathsf{content}_\tau(t)$ is finite for eveery $t \in \phi(A)$. Monotonic operators having finite content are necessarily continuous, but in the absence of something like the axiom of dependent choices, the converse does not seem to hold (cf. the proof of theorem 3 below). A finitary operator necessarily has finite content, but the converse certainly need not hold. For example, the equation:

$$\phi(X) = \{x : X|X\text{ is uncountable}\} \tag{34}$$

defines an operator which has finite content but is not finitary.

We now record the main results of this section:

Theorem 1 *ϕ_τ is a finitary (and hence continuous) operator and is a subfunctor of τ in the sense of section 3.*

Proof: Straightforward from the definitions using lemma 5. □

Theorem 2 *If the operator ϕ is typable with type τ and has finite content, then for any X, $\phi(X) \subseteq \phi_\tau(X)$. In particular, this holds for any finitary operator ϕ.*

Proof: Straightforward from the definitions using lemma 5. □

Our final result in this section requires the axiom of dependent choices, or at least, by a slight generalisation of the proof below, the following, apparently weaker, statement: *if X is an infinite set then X has a countable infinite family X_i of (pairwise distinct) finite subsets.*

Theorem 3 (DC) *If the continuous operator ϕ is typable with type τ, then for any X, $\phi(X) \subseteq \phi_\tau(X)$.*

Proof: We have to show that every $A \in \phi(X)$ has $\mathsf{content}^X_\tau(A)$ finite. Assume for a contradiction that some $A \in \phi(X)$ has $\mathsf{content}^X_\tau(A)$ infinite. Appealing to DC, let $\{x_0, x_1, \dots\}$, where the x_i are pairwise distinct, be a countably infinite subset of A and define a chain, $X_0 \subseteq X_1 \subseteq \dots$, of subsets of X as follows:

$$\begin{aligned} X_0 &= X \setminus \{x_0, x_1, \dots\} && (35)\\ X_{i+1} &= X \cup \{x_i\} && (36) \end{aligned}$$

Then $X = \bigcup_i X_i$, and so, as ϕ is continuous, $\phi(X) = \bigcup_i \phi(X_i)$, whence $A \in \phi(X_k) \subseteq \tau(X_k)$ for some k. But this implies that $\mathsf{content}^X_\tau(A) = \mathsf{content}^{X_k}_\tau(A) \in \mathbb{P}(X_k)$ which is impossible since $x_k \in \mathsf{content}^X_\tau(A)$ and $x_k \notin X_k$. □

5 Fixed Points of the Maximal Typable Operators

In this section, we construct fixed points for the maximal typable operators ϕ_τ defined in the previous section using only the Zermelo axioms without choice (or even dependent choices). In fact, the construction will work for an monotonic operator of type τ, having finite content that is a subfunctor of τ — thus the arguments of section 4 are only required to handle operators that are not functorial. The construction amounts to an encoding of the standard construction of a fixed point for a continuous operator from below as given in lemma 1.

For completeness, we devote a subsection to the construction of certain sets of finite trees that are required for the encoding. The construction of these trees is quite standard and readers who wish to skip the details are invited just to skim the terminology set up in theorems 4 and 5 in subsection 5.1 below and then move on to the main construction in subsection 5.2.

5.1 Trees

Theorem 4 *There is a typable operator $\mathcal{T}(X)$ such that for any set A, $\mathcal{T}(A)$, which we call the set of* finite trees with labels in A, *can be equipped with functions:*

$$\begin{aligned} \mathsf{Node} &: A \times \mathsf{seq}\,\mathcal{T}(A) \to \mathcal{T}(A) && (37)\\ \mathsf{Label} &: \mathcal{T}(A) \to A && (38)\\ \mathsf{Children} &: \mathcal{T}(A) \to \mathsf{seq}\,\mathcal{T}(A) && (39) \end{aligned}$$

enjoying the following properties:

1. *(Monotonicity) For $B \subseteq A$, $\mathcal{T}(B) \subseteq \mathcal{T}(A)$.*
2. *(Induction Principle) $\mathcal{T}(A)$ is the smallest set closed under formation of new trees from old using* Node, *i.e., if $B \subseteq A$ is such that $\mathsf{Node}(b, <t_1, \ldots, t_k>) \in B$ whenever $b \in B$ and $b_i \in \mathcal{T}(B)$ $(1 \le i \le k)$, then $B = A$.*
3. *(Constructor/Destructor Principle) For any $a \in A$, $t \in \mathcal{T}(A)$, $t_i \in \mathcal{T}(A)$, we have:*

$$\begin{aligned} \mathsf{Label}(\mathsf{Node}(a, <t_1, \ldots, t_k>)) &= a && (40)\\ \mathsf{Children}(\mathsf{Node}(a, <t_1, \ldots, t_k>)) &= <t_1, \ldots, t_k> && (41)\\ \mathsf{Node}(\mathsf{Label}(t), \mathsf{Children}(t)) &= t && (42) \end{aligned}$$

4. *(Recursion Principle) For any set C, and function $d : A \times \mathsf{seq}\,C \times \mathcal{T}(A) \to C$, there exists a unique function $f : \mathcal{T}(A) \to C$ satisfying the following equation:*

$$f(\mathsf{Node}(a, <t_1, \ldots, t_k>)) = d(a, <f(t_1), \ldots, f(t_k)>, \mathsf{Node}(a, <t_1, \ldots, t_k>)) \quad (43)$$

Proof: For brevity, we simply describe the construction and sketch what has to be proved.

The first two properties listed above only involve the function Node. If we can find an operator, $\mathcal{T}(X)$, and a *bijection,* Node, enjoying monotonicity and the induction principle, then we can use the equations in the constructor/destructor principle to define functions Label and Children as required and we can deduce the recursion principle by adapting a well-known argument for justifying definition by recursion over the natural numbers (e.g., see [3]). So it suffices to define $\mathcal{T}(X)$ and Node and show monotonicity and the induction principle.

We describe the construction of $\mathcal{T}(A)$, for a concrete set A, leaving it to the reader to check that the construction can be captured in a typable operator $\mathcal{T}(X)$.

To define $\mathcal{T}(A)$, we first construct the set, $\mathcal{T}$, of unlabelled finite trees. This is done by a standard construction, e.g., see [2]. The idea is that we represent a finite tree as a set of addresses for the nodes in the tree. The addresses are sequences of positive natural numbers. The root has the empty sequence, $<>$, for its address; a non-empty sequence, $<n_1, \dots, n_k>$, represents the node you arrive at by a k-stage journey starting from the root and following the n_i-th branch at stage i. Clearly some finite sets of sequences will not represent the set of node addresses of any tree. To filter out the non-trees, we define $\mathcal{T}$ in stages[4], $\mathcal{T}_0, \mathcal{T}_1, \dots$, with $\mathcal{T}_i$ comprising all trees of depth at most i. Each set $\mathcal{T}_i$ is a subset of $\mathbb{P}(\mathsf{seq}(\mathbb{N}_1))$, where $\mathbb{N}_1$ denotes the set of positive natural numbers. Formally, for $i \in \mathbb{N}$, we define:

$$\begin{aligned} \mathcal{T}_0 &= \{\} && (44)\\ \mathcal{T}_{i+1} &= \mathcal{T}_i \cup \{S \mid \exists n : \mathbb{N}_1 \bullet \exists s : \mathsf{seq}(\mathcal{T}_i) \bullet S = \{<>\} \cup \{<i> \frown u \mid i \in \mathsf{ran}\, s \land u \in s_i\}\} && (45)\\ \mathcal{T} &= \bigcup_i \mathcal{T}_i && (46) \end{aligned}$$

We have defined our unlabelled trees to be sets of node addresses. A labelled tree can then comprise a function from the set of node addresses representing an unlabelled tree to the set of labels. More formally, the set $\mathcal{T}(A)$ is defined to be the set of all functions $t : S \to A$ where $S \in \mathcal{T}$. The function Node is defined by the equations:

$$\begin{aligned} \mathsf{Node}(a, <t_1, \dots, t_k>)(<>) &= a && (47)\\ \mathsf{Node}(a, <t_1, \dots, t_k>)(<i> \frown s) &= t_i(s) && (48) \end{aligned}$$

where $a \in A$, $1 \leq i \leq k$, $t_i \in \mathcal{T}(A)$ and $s \in \mathsf{dom}\, t_i$. The domain of Node is the tree comprising $<>$ and all sequences $<i> \frown s$ for which the right-hand side of the second equation above is defined. It can now be verified that Node is monotonic, bijective and enjoys the principle of induction. By the observations at the beginning of the proof, this is sufficient to prove the theorem.

□

In the main construction in section 5.2, we will need the operator Hered defined in the following theorem. Given a subset, V, of the set $\mathcal{T}(A)$ of trees with labels in A, $\mathsf{Hered}(V)$ comprises those trees which are *hereditarily* members of V.

Theorem 5 *There is an operator,* Hered, *with the following property: if A is any set, and V is any subset of the set $\mathcal{T}(A)$ then $\mathsf{Hered}(V)$ is the (unique) subset of $\mathcal{T}(A)$ such that:*

$$\mathsf{Hered}(V) = \{t : \mathcal{T}(A) \mid t \in V \land \forall c : \mathsf{ran}(\mathsf{Children}(t)) \bullet c \in \mathsf{Hered}(V)\} \qquad (49)$$

[4] An alternative approach, followed in [2], is to define a partial ordering on sequences of positive natural numbers, corresponding to the order in which nodes are encountered in an appropriate traversal of a tree, and then to define a tree to be any set of sequences that is downwards closed with respect to that partial ordering.

Proof: The proof is an exercise in using the recursion principle of theorem 5. Let us define a function $d : A \times \mathsf{seq}\,\mathbb{N} \times \mathcal{T}(A) \to \mathbb{N}$ as follows:

$$d(a, <n_1, \ldots, n_k>, t) \quad = \quad \begin{cases} 0 & \text{if } t \in V \text{ and } n_1 = \ldots = n_k = 0 \\ 1 & \text{otherwise} \end{cases} \tag{50}$$

Given this choice of d, the recursion principle provides us with a function $f : \mathcal{T}(A) \to \mathbb{N}$ such that $f(t) = 0$ iff. $t \in V$ and $c \in V$ for each child, c, of t. Taking $\mathsf{Hered}(V) = f^{-1}\{0\}$ completes the proof.

□

5.2 The Main Construction

Throughout this section, ϕ will denote a typable operator, of some type, τ. ϕ will be assumed to be monotonic, to have finite content, and to be a subfunctor of τ. We will show that any such ϕ possesses a fixed point on the basis of the axioms of Zermelo set theory without using any form of the axiom of choice. Since the maximal typable operators subsume any monotonic operator of finite content by theorem 2 and satisfy our hypotheses on ϕ by theorem 3, this shows that any such operator possesses a fixed point. If we admit the axiom of dependent choices, theorem 3 gives us fixed points for all typable continuous operators.

Our construction is motivated by the construction in ZF of a fixed point from below for ϕ as discussed in section 2. Since our ϕ is monotonic and has finite content, and so is continuous, we only need to continue the construction up to the first infinite ordinal, i.e, we form a countable chain of sets as follows:

$$T_0 \quad = \quad \emptyset \tag{51}$$

$$T_{i+1} \quad = \quad \phi(T_i) \tag{52}$$

Note here that the monotonicity of ϕ and an easy induction show that $T_i \subseteq \phi(T_i)$ for every i. As we remarked in section 2, the union of the sets T_i will provide a fixed point for ϕ, but cannot be formed without an appeal to the axiom of replacement. We will circumvent this problem by encoding members of $\bigcup_i T_i$ as members of a set that we can construct without replacement. In outline, the encoding works as follows: suppose an element t of $\bigcup_i T_i$ is in T_{n+1}, say, but not in T_n; assume that we have already encoded all members of T_i for $i \leq n$; by our assumptions on ϕ, t, which is a member of $T_{n+1} = \phi(T_n)$, must be a member of $\phi(A)$ for some finite subset A of T_n. To encode t, we think of it a τ-tree with labels in A, as discussed in section 2. Choosing some enumeration, $A = \{a_1, \ldots, a_k\}$, of A, we encode t as a tree in the sense of section 5.1. This tree has for its root label the object obtained from t by replacing each leaf a_i in the τ-tree t with the number i. Its children are the encodings of $a_1, \ldots, a_k$. Thus the set of encodings of members of $\bigcup_i T_i$ will be of a subset of the set $\mathcal{T}(\phi(\mathbb{N}_1))$ of trees labelled by elements of $\phi(\mathbb{N}_1)$.

To carry out the encoding as just described would require the axiom of choice to pick an enumeration of the set A. Moreover, it would be difficult to describe the image of the encoding in $\mathcal{T}(\phi(\mathbb{N}_1))$. To avoid these problems we give a uniform construction, normalising the space of encodings so that no arbitrary choices are required. We describe the construction by a process of successive approximation, the third and final approximation providing the exact answer.

Our first approximation, $\mathcal{S}_1$, identifies those elements of $\mathcal{T}(\phi(\mathbb{N}_1))$ that can sensibly be decoded to give members of $\bigcup_i T_i$. Of course, the ZF axioms would be required to describe the decoding process, but in Zermelo we can at least say what it means to be decodable:

$$\mathcal{S}_1 \quad = \quad \mathsf{Hered}\{t : \mathcal{T}(\phi(\mathbb{N}_1)) | \mathsf{content}_\tau(\mathsf{Label}(t)) = \mathsf{dom}(\mathsf{Children}(t))\} \tag{53}$$

Recalling the definition of the operator Hered from theorem 5, it should be clear how, in ZF, we could decode a member, t, of $\mathcal{S}_1$ as a member of $\bigcup_i T_i$, by replacing elements of $\mathsf{content}_\tau(\mathsf{Label}(t))$ by the corresponding decoded member of $\mathsf{Children}(t)$. However, this decoding would be many-to-one, because $\mathcal{S}_1$ contains many redundant encodings of any given member of $\bigcup_i T_i$, and that will not do. Our next approximation will begin to eliminate this redundancy. To define it, we need a notion of isomorphism for members of $\mathcal{S}_1$: let us say that members t and u of $\mathcal{S}_1$ are isomorphic and let us write $t \sim u$ iff. there is a bijection f between $\mathsf{content}_\tau(\mathsf{Label}(t))$ and $\mathsf{content}_\tau(\mathsf{Label}(u))$ such that $\hat{\tau}(f)(\mathsf{Label}(t)) = \hat{\tau}(\mathsf{Label}(u))$ and for each i in $\mathsf{content}_\tau(\mathsf{Label}(t))$, $\mathsf{Children}(t)_i \sim \mathsf{Children}(u)_{f(i)}$. That is to say, thinking of $\phi(\mathbb{N}_1)$ as a set of τ-trees with numerical labels at the leaves, $t \sim u$ means that $\mathsf{Label}(t)$ and $\mathsf{Label}(u)$ are structurally identical modulo a permutation of their leaf labels that relates each child of t to an isomorphic child of u. We now define $\mathcal{S}_2$ as follows:

$$\mathcal{S}_2 \quad = \quad \mathsf{Hered}\{t : \mathcal{S}_1 | \forall i, j : \mathsf{dom}(\mathsf{Children}(t)) \bullet i \neq j \implies \mathsf{Children}(t)_i \not\sim \mathsf{Children}(t)_j\} \tag{54}$$

However, this definition of $\mathcal{S}_2$ still fails to make the decoding process one-to-one because a member t of $\mathcal{S}_2$ could contain two distinct but isomorphic subtrees that are not fellow siblings. Even if such situations were eliminated, reordering the children of t and relabelling its root label will give diverse encodings for the tree that t encodes. To solve both these problems, we take equivalence classes modulo $\sim$:

$$\begin{aligned} \mathcal{S}_3 &= \mathcal{S}_2/\sim & (55)\\ &= \{A \subseteq \mathcal{S}_2 | \exists t : \mathcal{S}_2 \bullet \forall u : \mathcal{S}_2 \bullet u \in A \iff u \sim t\} & (56) \end{aligned}$$

We now claim that $\mathcal{S}_3$ furnishes the fixed point that we need. By lemma 2, it suffices to exhibit an injection, κ say, from $\phi(\mathcal{S}_3)$ into $\mathcal{S}_3$. To define κ, let x be an element of $\phi(\mathcal{S}_3)$. By our assumptions on ϕ, $\mathsf{content}_\tau(x)$ is a finite subset, A, of $\mathcal{S}_3$. I.e., for some n, there exists a bijection f between A and $1 \mathinner{..} n$. Now, thinking of x as a τ-tree with labels drawn from A, $\kappa(x)$ will be the set of all trees whose root label can be got by using such a bijection f to relabel the leaves of x and whose children can be got by choosing representatives in $\mathcal{S}_2$ for the elements $f^{-1}(1), \ldots, f^{-1}(n)$ of $\mathcal{S}_3$. The following lemma puts this construction more formally:

Lemma 6 *The equation:*

$$\begin{aligned}\kappa(x) = \{t : \mathcal{S}_2 \quad | \quad & \exists n : \omega \bullet \exists f : \mathsf{content}_\tau(x) \rightarrowtail\!\!\!\!\rightarrow 1 \mathinner{..} n \bullet \exists s : \mathsf{seq}\, \mathcal{S}_2 \bullet \\ & \mathsf{dom}\, s = 1 \mathinner{..} n \wedge \\ & (\forall i : 1 \mathinner{..} n \bullet s_i \in f^{-1}(i)) \wedge \\ & t = \mathsf{Node}(\hat{\phi}(f)(x), s)\}\end{aligned} \tag{57}$$

defines an injective function κ *from* $\phi(\mathcal{S}_3)$ *to* $\mathcal{S}_3$.

Proof: The proof is routine, and we just give a sketch. First we must show that (57) does indeed define a function from $\phi(\mathcal{S}_3)$ to $\mathcal{S}_3$. This can be verified from the definitions, using the fact that ϕ is a subfunctor of τ to show that the variable t does indeed range over elements of $\mathcal{S}_2$. The definition of $\sim$ allows us to conclude that the right-hand side of the equation is actually an equivalence class of $\sim$ and so a member of $\mathcal{S}_3$. Now we must show that κ is injective. This follows from the injectivity of Node and the fact that $\hat{\phi}$ sends injections to injections.

□

Theorem 6 *If the operator* ϕ *is typable with type* τ *and has finite content, then for some subset* Y *of* $\mathbb{P}(\mathcal{T}(\tau(\omega)))$, $Y \simeq_{\mathsf{LFP}} \phi(Y)$. *In particular, this holds for any finitary operator* ϕ.

Proof: By theorem 1 and lemma 6, the maximal operator ϕ_τ has a fixed point. By theorem 2, $\phi(X) \subseteq \phi_\tau(X)$ for every X, so that the fixed point for ϕ_τ yields a least fixed point for ϕ by lemma 2. □

Theorem 7 (DC) *If the continuous operator* ϕ *is typable with type* τ*, then for some subset* Y *of* $\mathbb{P}(\mathcal{T}(\tau(\omega)))$, $Y \simeq_{\mathsf{LFP}} \phi(Y)$.

Proof: The proof is just like that of theorem 6 using theorem 3 insteand of theorem 2. □

6 Conclusions

We have strengthened the results of [1] to show, without recourse to the axioms of choice and replacement, that finitary typable operators theory have least fixed points. In fact, this holds of a somewhat wider class of operators: the monotonic typable operators having finite content. As usual, avoiding the axiom of choice gives a more constructive proof.

Using the axiom of dependent choices — generally considered to be a much more palatable axiom than the full axiom of choice — continuous typable operators have also been dealt with. The use of axiom of dependent choices or some similar principle seems to be essential, although we have been unable to give examples to demonstrate this.

There is much interest in mechanizing proof for languages such as Z. At the very least, the results in this paper provide a basis for consistency proof obligations in a tool supporting specification and proof in Z. For those who wish to follow a very rigorous line, the main constructions of this paper should be implementable within a proof tool such as HOL, Isabelle, or ProofPower. This would allow the defining properties of a recursive type definition to be derived rather than introduced as axioms. In a sense, this would provide a half-way house between the approach of Melham [4], whose work in HOL covers a limited class of operators within a typed set theory, and that of Paulson, whose work in Isabelle-ZF implements the standard ZF constructions in untyped set theory.

Acknowledgments

I am indebted to Randy Dougherty of Ohio State University for providing the key idea behind the construction in section 5.2.

References

[1] R.D. Arthan. On Free Type Definitions in Z. In J.E. Nicholls, editor, *Z User Workshop, York 1991*. Springer-Verlag, 1992.

[2] H.P. Barendregt. *The Lambda Calculus*, volume 103 of *Studies in Logic and the Foundations of Mathematics*. North Holland, 1984.

[3] Paul R. Halmos. *Naive Set Theory*. Springer-Verlag, 1974.

[4] Thomas F. Melham. Automating Recursive Type Definitions in Higher Order Logic. In G. Birtwistle and P. A. Subrahmanyam, editors, *Current Trends in Hardware Verification and Automated Theorem Proving*. Springer-Verlag, 1989.

[5] J.M. Spivey. *The Z Notation: A Reference Manual*. Prentice-Hall, 1989.

[6] J.M. Spivey. *The Z Notation: A Reference Manual, Second Edition*. Prentice-Hall, 1992.