SYMBOLIC SETS AND THE REAL LINE

MAURICE KARNAUGH

Abstract- This tutorial is intended for students of the sciences or engineering who are interested in theory and who would like a relatively brief, intuitively accessible introduction to set theory that meets their limited needs. The present mainstream axiomatic set theory is based upon the Zermelo-Fraenkel axioms (ZF) or ZF plus the axiom of choice (ZFC). This theory permits the creation of unimaginably large sets, populated by mostly undefinable elements. The mathematical "real line" is such a set.

The role of scientific theory in the scientific enterprise has been to provide symbolic models for real phenomena. There seems to be no foreseeable need for elements that are too numerous to be defined, or even named, in such work. Symbolic sets are well defined sets of symbolic elements which we are able to construct with the aid of suitably constrained axioms. The definate (pronounced de-fine'-ate) real line contains just the well defined elements of the real line and it is sufficient for numerical scientific computation.

ZF and ZFC were influenced by the development of formal logic in the 19th century. This paper takes a constructive point of view that makes it less abstract and more accessible and is influenced by the development of computer science.

An acquaintance with discrete mathematics and with convergent sequences is probably all that one needs in order to study this paper because we shall define the concepts we introduce. Further definition and discussion of any standard mathematical terms used herein can easily be found through a keyword search of the web.

Table of Contents

Contents

1. Introduction

Throughout much of human history, progress in mathematics has been motivated by practical concerns, largely in such matters as agriculture, commerce, technology and science. This circumstance began to change when progress in the study of logic led to the formalization of deductive proofs. During the late nineteenth century,

a number of mathematicians and logicians became actively engaged in attempts to give mathematics a firm foundation in formal logic. One very ambitious goal was, having defined the logical language and some consistent mathematical axioms, to then deduce all that was then known of mathematics. Unfortunately, it is unclear precisely which axioms should be employed and not possible to prove their consistency. Logical foundations for mathematics are presently not well established but a large body of interesting mathematical theory has resulted. (For a survey of the philosophical foundations of mathematics, we suggest the online Wikipedia article on *Foundations of Mathematics.*)

An important result of this work has been the divergence between mathematical set theory and the applications of mathematics to scientific theory. Much of this divergence can be ascribed to differences in method as it applies to the selection of axioms. The mathematical logicist view is that one can be free to introduce any axiom that is sufficiently interesting and which (one believes) cannot be proved or disproved by the preexisting axiom set. Having done that, one can then proceed to deduce some of its consequences, however counterintuitive they may be.

The scientific method is much more constrained. This takes some explaining. Up to this time, the goal of scientific theory has been to develop symbolic theories that adequately describe the observable properties of real phenomena. These properties are determined by observation and, when possible, by conducting experiments. If one chooses axioms that are too simple or unnecessarily broad in scope, one increases the chance that in the near future experiments will reveal some discrepancies of theory with observation. A common view among scientists is that a theory should be as simple as possible **but not more so**.

Simplicity alone is an inadequate criterion. One might argue that Newtonian physics is simpler than general relativity but general relativity is more correct in predicting certain critical observations. One's intuition is also not invariably a correct guide. Einstein, for example, found it difficult to accept quantum theory because he believed that "God does not play at dice with the world." Nevertheless, quantum theory is the most accurate model for many physical observations. To sum up these simple remarks, nature is the arbiter of the selection of natural laws - not simplicity, not interestingness, not intuition.

One more thing: theoretical models must be communicable by means of texts in a sufficiently expressive language. The same thing is also true of logic and mathematics but logicism has introduced some new philosophic problems. For example, in the standard form of modern set theory, sets of elements can be so large that most of their elements cannot be defined by any text. In addition, the axiom of choice tells us that these undefinable elements can be well ordered (a specific type of order

which will be explained in this paper). These counterintuitive things need not be dismissed forever. However unlikely it may seem, scientists may someday be shocked to discover that they need ridiculously large collections of well ordered undefinable elements.

Meanwhile, because of a lack of foreseeable demand for them in the present scientific marketplace, we may be excused for tentatively rejecting undefinable elements and, instead, examining those interesting ideas that set theory has developed which are useful for a theory of symbolic sets. This can be done by modifying the set theoretic axioms. In doing this, it must be acknowledged that much of modern mathematics presently depends upon the standard axioms, including the axiom of choice, and, in its present form, is beyond the scope of symbolic set theory.

Two external references are indicated by single integers in square brackets and can be found in the References section. Most of our our external references will be sequences of keywords within square brackets, used as needed. For example, the Wikipedia article on *Foundations of Mathematics* could be referenced as [wikipedia foundations mathematics]. Search for it without the brackets. A good search engine will return that article as an early choice. An article on the early development of set theory can provide a more complete context for this introduction [early development set theory].

Internal references, if used, will be decimals (eg. [3.2] points to the second numbered item in section 3). [3.0] points to section 3 but not to any numbered item in it. Two textbooks are referenced, [1] and [2], for those who desire to study ZFC [ZermeloFraenkel set theory]. The first one is brief but quite abstract. The second one is also abstract but much more comprehensive and well designed for lengthy self study. Our approach is much less abstract than the textbooks'.

This printed tutorial is in a language we call a **meta-language**. The alphabet of this meta-language includes three fonts of the English alphabet and a large collection of punctuation, formatting and logical and mathematical characters. This is a large but finite alphabet. The existence of a sufficiently expressive meta-language is esssential for the practice of philosophy, science, mathematics and for this paper. It is usually taken for granted but we must be careful to observe its limitations.

Our method is largely constructive, based upon recursive generators. The presentation may be described as informal, minimally abstract and strongly influenced by computer science.

2. Characters and Strings

Characters are visually distinguishable printed objects. A collection of distinguishable characters is called a class of characters. If that class of characters has certain required properties it can constitute an alphabet. Each character in an alphabet is also called an element of it and appears only once in it.

Examples of characters would normally be the elements of some font, or several fonts, such as English letters, numerals, and various other distinguishable shapes which might be employed in logical expressions, mathematical expressions, punctuation, or formatting. A non-empty class of characters might contain just one or finitely many or even infinitely many distinct characters. We shall see examples of that. What we call a character might better be called a character type because the same character might appear multiple times in some string.

Strings are non-empty sequences of characters that always have a first character. Finite strings also have a last character while infinite strings have no last character. A given character may appear more than once in a string. Each string will be displayed as a row of characters, the first one being the leftmost and the last one is the rightmost. x3y2z3 is an example of a finite string. Order of the characters is important. Two strings are equal if and only if they are of the same length and their character sequences are pairwise identical. Special characters, such as for punctuation, empty spaces and carriage returns may be included in an alphabet and may occur in strings.

When we say that a string exists, we mean that we can identify the first character, the second character, the third character and so on up to the last character (if the string is finite). A finite string of characters in some alphabet will be called a **symbol** of that alphabet, or a **text** of that alphabet. The difference between a symbol and a text is that a symbol may be any finite string of characters of a given alphabet while a text is assumed to be meaningful. Texts constitute a subclass of symbols. Therefore, the class of all texts of a given language cannot have more elements than the class of all symbols of that language.

While we will have some use for infinite strings, which have a first character but no last character, the terms symbol and text will always mean a finite string of characters.

A collection of symbols will be called a class of symbols. If a symbol x is in a class Z, then x is said to be an element of Z or a member of Z and Z is said to contain x.

We shall not be concerned with the problems of linguistics, such as grammars, semantics, or meaningfulness, but we assume the existence of some meta-language in which such meaningful things as definitions, axioms, theorems, algorithms, etc. and pedagogical discussion can be written. Simply put, all communicable knowledge of mathematics can be expressed by texts in such a language.

3. Elements and Classes

Many of the mathematical objects we shall be talking about will generically be called elements. A class is a collection of distinct elements, usually with some common properties that make the class interesting. Classes and elements are connected by the **membership** relation. If x represents an element and Y represents a class, then $x \in Y$ means that x is a member of Y and $x \notin Y$ means that x is not a member of Y. If x is a member of Y then Y is said to **contain** x. When speaking of classes we shall assume (unless otherwise stated) that the classes in question all contain elements that are contained in some well defined largest class which we shall call **the universe of discourse**, a term which we shall normally abbreviate to **the universe**.

Axiom of membership 3.1. *If x is an element and Y is a class, then either $x \in Y$ or $x \notin Y$ but not both.*

We shall carefully observe the distinction between elements and classes. None of the classes we shall define is an element. Such classes are called **proper classes**. For ease of reading, but without any other mathematical significance, the elements of a class may, optionally, be listed in parentheses, for example, the class (a, b, c). The empty class, (), requires parentheses.

The collection of all elements that are members of a class is called the **extension** of that class. The extension of a class is an important property of that class from which other properties can be deduced, for example, its **cardinality**, by which we mean the number of its elements. Each element in the extension of a class is unique and is counted only once. A class may not contain multiple occurrences of any element.

Definition of class equality. *Two unordered classes are equal if and only if they have the same extension.*

The order in which we may list the elements of a class is irrelevant unless a specific order has been assigned. Class (b, c, a) is equal to class (a, b, c). All empty classes are equal because they have the same extension. Ordered classes will be displayed

within angle brackets. For example $< a, b, c >$ is a class to which the displayed order has been assigned and $< b, c, a >$ is not equal to $< a, b, c >$.

Certain operations are defined on classes. These will be represented in functional forms rather than in operator notation (with the exception of some very familiar operations such as the operations of ordinary arithmetic). If X and Y are classes then $union(X, Y)$ is a function that returns a class containing only every element that is a member of X or a member of Y. If an element occurs in both X and Y it nevertheless occurs only once in $union(X, Y)$.

A collection of classes will be called a family of classes. It cannot be called a class because classes are not elements. If z is a family of classes then $union(z)$ is a class containing only every element that is a member of some class in z.

Theorem of unions 3.2. *The union of any family of classes exists and is a class.*

Proof. This follows immediately from the definition of union and the definition of classes. □

If X and Y are classes then $intersection(X, Y)$ is a function that returns a class containing only every element that is a member of both X and Y . If z is a family of classes then $intersection(z)$ is a class containing only every element that is a member of every class in z.

If X and Y are classes then the function, $difference$<X, Y>, returns a class containing only every element that is a member of X and not a member of Y. The order of the arguments is relevant in this function, therefore they are listed within angle brackets instead of parentheses. We shall adhere to this practice.

We do not require existence axioms for the union, the intersection or the difference of classes because their existence follows from the definitions.

There are also class inclusion relations. If P and Q are classes then $P \subseteq Q$ means that every member of P is also a member of Q. This is also expressed as P is a **subclass** of Q or as Q **includes** P. $P = Q$ if and only if $P \subseteq Q$ and $Q \subseteq P$. When $P \subseteq Q$ but it is not true that $Q \subseteq P$, which is expressed as $Q \subsetneq P$, then we say that P is a **proper subclass** of Q, which is expressed as $P \subset Q$. The empty class is a class and is a proper subclass of every other class.

It follows from the definition of classes that every subclass of a class is also a class.

Observe that if X is a class then $X \subseteq X$ and $() \subseteq X$.

Classes may have properties associated with their construction or their extensions, such as an ordinal number and a cardinal number. These will be discussed later. Many of the properties and relations we attribute here to classes are also commonly attributed to sets. Classes and sets are both collections of elements but we find it convenient to deal with classes first.

4. Generating Alphabets

In order to create an **alphabet** we shall employ a few distinguishable **primal** printed objects to begin with. Then we can use some rules for putting them together in order to assemble our characters. We shall create strings of primals to form alphabetical characters.

How many primals must we have? It is clear that we shall need at least one because an empty class which can construct only the empty string will not be sufficiently useful. Suppose we have just the primal object, 1. We can then introduce an operation called **concatenation** to produce strings. For example, we can concatenate a 1 to another 1 to produce the string 11. We can then concatenate another 1 to that and get 111, and so on indefinitely. We can also concatenate 11 to 111 to get 11111.

Strings of just one primal printed object cannot be a satisfactory alphabet because of the **segmentation** problem. If we are given the string 11111 we cannot tell whether it is the concatenation of 1 with 1111 or the concatenation of 11 with 111 and we cannot tell whether the shorter part was concatenated to the left or the right of the longer part. Concatenation of such strings results in a loss of information.

The segmentation problem can be solved by having at least two primal printed objects, for example, 0 and 1. Using these two primals we can create characters like 0, 01, 011, 0111, and so on. Concatenation of these characters loses no information because each character begins with a 0. For example, the right concatenation of 0111 to 011 is clearly 0110111. The left concatenation of 0111 to 011 would yield 0111011.

A class of characters which can be concatenated without loss of information will be called **segmented**. **We require that every alphabet be segmented.**

It is possible to create non-empty segmented alphabets of arbitrarily many characters or even a class of infinitely many characters from which alphabets can be selected. The creation of infinitely many characters can be done by a process we call **simple finite recursion**. The following example illustrates simple finite recursion.

Every simple finite recursion procedure begins with a single element called the **foundation** element or with a finite sequence of elements called the **foundation sequence**.

Begin with the foundation element, 0. The successor of 0 is 01. If x is any character, then the right concatenation of a 1 to x, which is $x1$, is the next character. $x1$ is called the **successor** of x. In functional terms, $suc(x) = concat{<}x, 1{>}$. This concatenation operation can be repeated (i.e., can **recur**) arbitrarily many times and the procedure is called simple recursion. This procedure is also called simple finite recursion because each element is generated after a finite number of recursions, even though there is no last element. A sequence of distinct elements with a first element but no last element will be called an **infinite** sequence.

The resulting infinite sequence, $0, 01, 011, 0111, \ldots$, will be called the **segmented unary** class, SU. It uses two primal elements but one of them is used only as a delimiter. Every 0 marks the start of a new character.

Had the successor function used left concatenation of a 1 rather than right concatenation, we would have gotten the segmented infinite class, $0, 10, 110, 1110, \ldots$, in which each 0 indicates the end of a character. This will be called the **unary segmented** class, US.

The usual binary form of the natural numbers, $0, 1, 10, 11, 100, \ldots$, is not segmented. In order to segment it, we would need to introduce a third primal object as a delimiter. We could, of course, use the first k characters of SU as primitives to generate arbitrarily large segmented alphabets. For example, the primitives, $0, 01, 011$, using 011 as a character delimiter, could generate a large segmented alphabet that includes the decimal digits as characters.

In generating infinite classes by means of simple finite recursion we have given them an additional property, **order**. Each character thus created has a unique successor. Of any two distinct characters, one must be created before the other. If x is created before y then x is said to be **less than** y. This is indicated by $x < y$. The class is said to be **totally ordered** by $<$. If x and y represent the same character, then we say that these characters are equal, $x = y$.

This total ordering relation, $<$, as we have defined it, is called a **strict ordering**. For any two **distinct** characters, x and y, either $x < y$ or $y < x$, **but not both**. Because the ordering is total it is called a **total strict ordering**.

There are two necessary conditions for a simple finite recursion to create an infinite class of elements. First, the successor function must be applicable to each element. If *suc* is applicable to element x, then it must be applicable also to the successor

element of x. That is, the property that *suc* is applicable must be **inherited** by all of the elements.

The second necessary condition is that each element so created must be unique. This is assured in our examples of SU and US because each character created is one primal longer than its predecessor. There can be no loop in the character sequence. This assures that the sequence created will be infinite and that $<$ has the **transitive** property: if x, y, and z are three elements such that $x < y$ and $y < z$ then $x < z$. Also, the transitive property assures that there cannot be a loop in the sequence of elements. When this is the case the elements of a total strict ordering are said to have a **linear** ordering.

A finite sequence can also be linearly ordered.

There is a special kind of linear ordering, called a **well ordering**.

Definition of well ordered 4.1. *A linearly ordered class is well ordered if and only if each non-empty subclass has a least element.*

In a well ordered class, the successor to an element x is the least element of the subclass of all elements that follow x.

Not every linearly ordered class of elements is well ordered. We shall see that the non-negative rational numbers are linearly ordered but they are not well ordered. We shall have much more to say about order relations.

We have seen that simple finite recursion permits us to construct infinite classes of characters from a finite class of primal characters. This is an important mathematical fact. It allows us to introduce the concept of infinity and also the concept of **completion**. By completion of a representation of the natural numbers, in this case by a recursion without end, we are able to say that the sum of any two natural numbers is another natural number. We call this the completion of the natural numbers with respect to addition. We shall adhere to this meaning of "completion."

Completion is no small matter. Although each element of a simple finite recursion is created after a finite number of *successor* operations, we must axiomatically assume that the completion of the recursion exists in order to assume that the class exists.

Axiom of completion. *The completion of a simple finite recursion is a class.*

5. Reification

Symbols are essential mathematical objects and they are routinely used in proofs without much comment about the method. For example, one might come across a proof that begins like this: *let x and y be positive integers.* What is going on here is that the symbols x and y have been selected to represent positive integers so that we can reason about them under the assumption that they have the properties of positive integers.

Unless other assumptions are specified, those are the only properties that may be used in the proof. In that case there is also no specification of precisely which positive integers x and y represent. In fact, the proof will require that they may represent any arbitrarily selected positive integers.

Let's call this process of naming mathematical objects **reification**. We shall use this term exactly as described here although it has been used in other ways elsewhere in philosophy, in literature, and even in bad jokes about "reified beans."

Also, we could specify some additional property in which case we have made a definite reification by giving x an additional property. For example, there is the

ZF Axiom of separation. *For any well defined class X there is a subclass Y consisting of all elements y of X for which $condY(x)$* holds.

In this formulation, $condY(x)$ may be expressed as a function whose domain is the elements of X and which returns 1 when a specified proposition about x is true (i.e., when it holds) and which returns 0 otherwise. This is a definite reification of Y, which is a subclass of X. Because all subclasses of X are also classes this axiom does not confer existence on the subclass but it definitely reifies it. What the definition of $condY(x)$ does accomplish is to establish that Y is a well defined subclass of X. We shall need a definition of **well defined**.

This definition will be addressed later. At this time we simply assert that any finite class of characters or symbols may be well defined by listing its elements and that any characters or symbols that can be generated by means of simple finite recursion are well defined and the entire class is well defined.

What we mean by *Y is well defined* is that $cond(Y)$ is defined by a text in some sufficiently expressive meta-language such that, given any element x of X, we can tell whether $condY(x) = 1$ or $condY(x) = 0$. In that case, we can also say that $condY$ is a well-defined algorithmic function. Functions of this type, which define a subclass, will be called **filters**. Filters are a kind of **algorithm**. Algorithms are a

kind of procedure which will return an answer and halt, while recursive generators need not halt.

Reification is an accepted mathematical method but it will be convenient to include it explicitly among the axioms of set theory. However, **we cannot reify more objects than we can symbolically name**.

Axiom of reification 5.1. *A mathematical object may be given a unique symbolic name so that we can reason about it.*

Reification is related to the definition of classes in an important way. What we mean by a non-empty class X is that we can reify an element x of it and we can form two new classes: one contains only the element that has been reified, (x), and the other contains its complement, $difference<X,(x)>$, with respect to the original non-empty class.

6. Concatenation Functions

The symbols we shall use are finite strings of alphabetical characters. The strings can be constructed by concatenations. We shall now define a family of concatenation functions that are sufficient for this purpose.

The simplest function is $concat<x,y>$ where x and y represent symbols. The return value of this function will be the concatenation of these symbols in the designated order, xy. This means that the first character of y is now the successor in xy of the last character in x. (Infinite strings cannot be concatenated because they have no last character, but a finite string can be left concatenated to an infinite string.)

If we want to represent the concatenation of several symbols we can use multiple arguments such as $concat<x,y,z>$ which will return xyz. The order of the arguments is significant. $concat<abc,def>= abcdef$ while $concat<def,abc>= defabc$. Therefore, the arguments are listed within angle brackets.

The use of multiple arguments facilitates a more concise notation. *concat* returns a symbol. It must have a finite linearly ordered class of symbolic arguments.

multicat is a very versatile function that takes a linearly ordered finite family of classes of symbols as arguments and returns a class of symbols. If X and Y are classes of symbols, $multicat<X,Y>$ returns the class of all concatenated symbols of the form xy where $x \in X$ and $y \in Y$. *multicat* can also take a larger finite list of arguments, of which some may be symbols and some may be classes of symbols. For

example, $multicat<X, y, Z>$ will return a class of all symbols of the form xyz, in which $x \in X$ and $z \in Z$ while y is a fixed symbol.

multicat can be implemented by an algorithm (i. e., a procedure that will terminate in a finite number of steps) when all of the classes are finite, but that is not true otherwise. Because the list of arguments is finite, *multicat* always returns a class of symbols. If one of the arguments is an infinite class then *multicat* will return an infinite class of symbols.

When we discuss the generation of symbols by classes, we shall see that the *multicat* of a finite, linearly ordered collection of linearly ordered classes of symbols returns a finitely generable class of symbols. In the case of an infinite argument, the multicat function is not algorithmic; it is a generator.

Another function, $finitecat(X)$ has a class of segmented symbols, X, as its domain and returns the class of all finite strings of elements of X. We shall see that this class is also generable when X is appropriately ordered but it is finitely generable only when X is finite. This is also the case when the input symbols are the characters of an alphabet. By restricting ourselves to finite alphabets, we can restrict the class of all symbols to symbols that are finitely generable.

Because the characters of any finite alphabet can be encoded as finite binary strings, all finite strings of such characters can be encoded as finite binary strings.

7. Generating Symbols and Classes

We have already seen that some infinite classes of characters can be generated by means of simple finite recursive generators. Let's examine two more classes of this kind.

Representations of the natural numbers, whether in binary, decimal, or any other positive integer number base, can be generated by using the successor function, **add1**, where the addition of 1 is carried out in a manner appropriate to that number base. Let x represent any element. If the number base is to be binary, then $add1(x) = x +_b 1$ and the addition is according to the rules of binary arithmetic. The foundation element is always 0 and the recursion has no end.

Another important example is the generation of all finite binary strings. Again, we use 0 as the foundation element. The successor function is a bit more complex. If x is a string of all ones, then $successor(x)$ is a string of all zeros that is one bit longer than x. Otherwise, $successor(x) = x +_b 1$, the binary successor. This procedure generates the class of all binary strings of length n, where n is any positive integer,

before going on to generate all binary strings of length n+1. Because each of the classes is finite, the entire class is finitely generated.

A similar procedure can be used to generate all finite strings of any finite alphabet containing 2 or more characters. Let's reify an alphabet of k characters by means of the symbols $x_0, x_1, x_2, \ldots, x_{k-1}$. These characters are ordered according to their subscripts, which are called **indices**. The $add1$ successor function for the indices is carried out in base k arithmetic. When the index string is a string of all $(k-1)$s, the next index string is a string of all 0's that is one digit longer.

Another general procedure for constructing all symbols of a given non-empty, finite alphabet will now be described. It is our first example of the use of class recursion. We shall call this procedure **canonical symbolic class generation**.

Let ABC represent any non-empty, finite, linearly ordered, segmented alphabet.

A concise representation for the class of all symbols of this alphabet is $finitecat(ABC)$, the class of all finite strings of characters of ABC. Let $multicat < X, Y >$ represent the class of all ordered concatenations of the form $concat < x, y >$ where x is any member of X and y is any member of Y. When X and Y are finite, $multicat$ is algorithmic but when one or both of them are infiinite we shall see that $multicat < X, Y >$ is finitely generable.

The recursive construction of $finitecat(ABC)$ is as follows.

The foundation class is the alphabet. $C_0 = ABC$, and the sequentially constructed classes will be indexed by natural numbers.

The recursion is $C_{k+1} = multicat{<}C_k, ABC{>}$. For example,

$$C_1 = multicat < C_0, ABC >= multicat < ABC, ABC > .$$

The kth class contains all strings of ABC characters of length $k+1$. There is no last class. The sequence of classes is infinite.

The union of all of these classes, $finitecat(ABC)$, is not the successor of any of them but it is a class. Such classes are called **limit classes**. The further concatenation of any two symbols in $finitecat(AB)$ cannot produce anything new because the result would be just a finite symbol. Therefore, this universe of symbols in ABC is said to be **complete** with respect to finite concatenations of symbols.

When the alphabet is finite the successor function is algorithmic and all the symbols are finitely generated. We assume that all alphabets are finite and linearly ordered as well as segmented. The ordering can be achieved merely by listing the characters.

It is noteworthy that, because all of the characters of such an alphabet can be encoded as distinct finite binary strings, all of the symbols of that alphabet can be encoded as distinct finite binary strings. Therefore, the class of all symbols of that alphabet cannot be more numerous than the class of distinct finite binary strings.

8. Cardinality

We need a method for quantifying the sizes of classes and for comparing them.

Let's begin with a procedure for **counting** things. Consider a non-empty class X of distinct elements. We do not know how many elements X contains so we employ the following procedure, which we shall call **countable reification**.

1) The foundation family. We create two classes: the initial residue class, $RX_0 = X$ and the initial cumulative class, $CX_0 = ()$, which is empty.

2) The successor operation is $count{<}RX_k, CX_k{>}$, where k is a representation of any natural number. This is a function whose argument is two natural number indexed classes which returns a family of two modified classes,
RX_{k+1} and CX_{k+1}. The modification of the residue class, RX_k is to reify one of its elements (it does not matter which one) and name it x_{k+1} and remove it from RX_k so that $RX_{k+1} = \ difference{<}\ RX_k, x_{k+1}{>}$. The modification of the cumulative class is $CX_{k+1} = \ union(CX_k, x_{k+1})$.

3) The recursion. The *count* function is applied recursively until, for some n, the residue class RX_n is empty.

If there is a natural number, n for which the recursion stops, then we say that the class X is finite and its **cardinality**, the number of its elements, is n.

If the recursion will never stop, then we say that X has **infinite** cardinality. We cannot further specify the cardinality of X but we can say that the linearly ordered class of reified elements that this process puts into CX_k for all natural numbers k is an infinite sequence. It has a first element but no last element. Furthermore, this sequence of elements has a clear **one-to-one relationship** with a subclass of the elements of X and it also has a clear one-to-one relationship with the class of natural numbers. It follows that every infinite class includes a subclass that has a one-to-one relationship with the natural numbers.

We call the cardinality of the natural numbers **denumerable**.

Definition of one-to-one relationship 8.1. *Two classes, X and Y, have a one-to-one relationship if and only if there is a function, $f(x)$, whose domain is all elements*

of X and whose range is all elements of Y such that each element of Y is the image under f of exactly one element of X and each element of X is the pre-image of exactly one element of Y.

Such functions are also called **bijections**. The inverse function, f^{-1}, of a bijection, f, is also a bijection. It is defined by $f^{-1}(f(x)) = x$ for each element x of X.

We shall abbreviate **one-to-one** as **1:1**.

If there is a bijection between classes X and Y, that is equivalent to saying that the elements of X and the elements of Y can be arranged in ordered pairs $< x, y >$ such that each x in X and each y in Y appears in exactly one pair.

It is clear that if two finite classes that have the same count, say n, both have a 1:1 relationship with the first n positive integers and, therefore, have a 1:1 relationship with each other. We extend the definition of equal cardinality to all classes as follows.

Definition of equal cardinality 8.2. *Two classes have equal cardinality if and only if there exists a 1:1 relationship between them.*

Two classes that have the same cardinality are also said to be **equipollent** or **equinumerous** or to have **equal power**. If two classes, X and Y, are equipollent then we say that $card(X) = card(Y)$, where $card$ represents the cardinality function and it returns a symbol that represents a cardinality.

All classes that are equipollent with some given class form an **equivalence family** with respect to cardinality. The equipollence of two classes, X and Y, is indicated by $X \approx Y$ and it is an **equivalence relation**. Any equivalence relation, $erel$, is **reflexive** ($XerelX$) and symmetric (if $XerelY$ then $YerelX$) and transitive (if $XerelY$ and $YerelZ$ then $XerelZ$). All classes that are equipollent with one class are equipollent with each other and form an equivalence family.

All classes that can be generated by means of simple finite recursions or by countable reification form an equivalence family with respect to cardinality. In particular, all such classes have a clear 1:1 relation with any representation of the natural numbers. Consider the following generic description of a simple finite recursive generator.

We begin with a foundation element. Let's reify it as x_0, We then apply an algorithmic successor function, *suc* to get the next element, x_1. This function is then applied recursively without end to get an infinite sequence of distinct elements, x_k, where k represents any natural number. Clearly, every class generated by this procedure will have the same cardinality, which is the cardinality of the natural numbers.

The cardinality of this equivalence family has been given the conventional name, $\aleph_0$ (pronounced **aleph null**). $\aleph$ is the first letter of the Hebrew alphabet.

The successor functions that can be employed in this fashion must be algorithmic, that is, the procedure must be completed by a finite sequence of finite steps. They must have the property that each element created by them must be a new element and not one of the previously generated elements. A sufficient condition for this will be introduced in a later section. Also, each element created must inherit the property of being within the domain of the sucessor function.

Any infinite subsequence of any infinite class that is created by the countable reification procedure also is 1:1 with the natural numbers and it also has cardinality $\aleph_0$.

Classes that have cardinality $\aleph_0$ are said to be **denumerable**. Classes that are either denumerable or finite are said to be **countable**.

How are we to compare the cardinalities of infinite classes? One reasonable approach is to extend what we know about finite classes but it must be done with care because, unlike finite classes, infinite classes can be in a 1:1 correspondence with some of their proper subclasses. For example the positive integers can be in a 1:1 correspondence with the even positive integers. In that case the bijection is $f(j) = 2j$. It is also clear that there is a bijection between the natural numbers and the positive integers. In this case the bijection is f(j)=(j+1).

We use the following rules of comparison.

Two classes are equipollent and have the same cardinality if and only if they can be placed in a 1:1 correspondence.

If a class B can be placed in a 1:1 correspondence with a subclass of a class C, then we say that C **dominates** B and write $B \preceq C$. This enables us to say that the cardinality of B is less than or equal to that of C. It follows that the cardinality of a subclass S of a class X is no greater than that of X.

If C dominates B but B does not dominate C then we say that C **strictly dominates** B, $B \prec C$, and that the cardinality of C is strictly greater than the cardinality of B.

If two sets are equipollent then each of them dominates the other.

If $B \preceq C$ and also $C \preceq B$ then it appears reasonable to equate their cardinalities, but in that case it should be possible to prove that there is a 1:1 relationship possible.

That mutual domination implies the existence of a 1:1 relation has been proved for all classes. This is commonly called the Schroeder-Bernstein theorem. The general proof will not be reproduced here but it can be found in the referenced textbooks or online [wikipedia Schroeder-Bernstein].

The cardinality of a class X will be denoted as a function, $card(X)$, from classes to cardinal numbers.

We have already seen that every infinite class has a subclass that is 1:1 with the class created from it by the process of countable reification and that this class has the same cardinality as the natural numbers, $\aleph_0$.

8.3 Because every infinite class has a subclass of cardinality $\aleph_0$, this is the smallest infinite cardinality.

It follows that every infinite class that is dominated by the natural numbers or the finite binary strings has cardinality $\aleph_0$. This applies to the class of all symbols of a finite alphabet and to all of its infinite subclasses.

9. Sets

Up to this point, all of the elements we have generated have been symbols. Now, we shall define another type of element called sets.

In the early days of set theory, sets were thought of as collections of elements, much as we have defined classes, but sets have one more very important property: sets are elements of a set theoretic universe and they can contain other sets as elements. This led to the discovery of contradictions within the theory. The most famous of these is Russell's paradox, which is discussed in the next section. This made necessary some revisions of set theory centered on restricting the creation of sets in order to avoid these contradictions. Consequently, various axioms of foundation or regularity have become part of set theory.

We shall employ such an axiom in a very simple form when we define the universes whose elements include sets. **Sets that meet the regularity condition are called well founded**. When we speak of sets it should be understood that we mean well founded sets within the universe of discourse.

The distinctions between classes and sets are these:
1) Any collection of elements within a universe is a class but, while sets are collections of elements, not every collection of elements can be the extension of a set.

2) Sets are elements but classes cannot be elements. Classes that cannot be elements are usually called **proper classes**. All of the classes we shall construct will be proper classes.

3) Because sets are elements, the extension of a set may include other sets.

In constructing sets from classes, we shall use a simple function that we call *wrap* which is not a standard term of ZF theory. It is easily illustrated in the case of finite classes and sets. If (a, b, c) is a regular class of elements then $\{a, b, c\}$ is a set. That is, $wrap(a, b, c) = \{a, b, c\}$. The inverse function is *unwrap*, so $unwrap\{a, b, c\} = (a, b, c)$. The term *wrap* is derived from the act of wrapping a collection of elements in curly brackets, which is a common notation for sets.

When dealing with infinite classes, this simple definition is inadequate. In general, the wrap of any class will be a set having exactly the same extension if and only if that class satisfies the condition of regularity. Classes that do not satisfy that condition cannot be wrapped. The *unwrap* of a set is always a class having the same extension.

Any class that satisfies the regularity condition will be called a **regular class**.

We shall also employ a function from a class to a class called *multiwrap*. If X is a regular class and all of its proper subclasses are also regular classes then $multiwrap(X)$ is the class containing $wrap(X)$ and the wraps of all of the proper subclasses of X as elements. For finite classes, the multiwrap is algorithmically computable. For infinite classes, the multiwrap exists axiomatically. We shall restrict this axiom later on.

The union of a family of classes is a class which contains only every element that is in any of them. The *wrap* of that class is a set if and only if that class is regular.

10. Russell's Paradox

In the early days of the twentieth century, the philosopher and logician Bertrand Russell, who was interested in placing mathematics on a firm logical foundation, was greatly troubled by a perceived paradox in set theory. At that time, it seems to have been thought that **the set of all sets** is a set that exists within the theory.

Russell's paradox is this. The set of all sets would have as a subset the set of all sets that do not contain themselves as elements. However, if we assume that this subset does not contain itself then it must; if we assume that it does contain itself then it must not. Clearly this presents an unavoidable contradiction also called a **paradox** or an **antinomy**.

The deduction of a paradox indicates that the axioms are inconsistent and need to be repaired in some fashion. In ZF set theory, as in our treatment, the axioms and the constructions employed preclude this embarrassment.

This problem makes it clear why the *wrap* function is needed. By limiting the domain of *wrap* to classes that are regular we can limit the sets of the universe to sets that are well founded and which avoid the paradox. Specific examples will be dealt with later.

11. Some Theorems About Cardinality

We now prove a number of useful theorems about cardinality.

Consider a denumerable family of classes, each of which contains a denumerable class of elements. We assume that no two of the classes have any element in common, that is, they are **disjoint**. What would be the cardinality of their union?

Theorem of denumerable unions 11.1. *The union of any denumerable family of pairwise disjoint denumerable classes is denumerable.*

Proof. The assumed classes are nameless and their elements are also nameless. Because the family of classes is denumerable, we can reify all of them by giving each of them the name of a natural number. We can also name each of the elements of any of these classes with a natural number. However, each of the names in the union must be unique. We can achieve this by using ordered pairs of natural numbers to name each of the elements of the union. For example, any such element can be named $< x, y >$, where x and y each represent a natural number. This name represents element y of class x.

Now let us use the unary segmented alphabet $US = (0, 10, 110, \dots)$ to represent the natural numbers. Because this alphabet is segmented, we can concatenate x and y without any loss of information; every 0 represents the end of a character. Therefore the names $< x, y >$ and $concat(x, y) = xy$ have a 1:1 relationship and both classes have a 1:1 relationship with the elements of the union. However, the elements xy are all finite binary strings and the class of all of them is an infinite subclass of the finite binary strings. We have already seen that the class of all finite binary strings is denumerable, so every infinite subclass of it is also denumerable. □

Every infinite class that is dominated by a denumerable class is also denumerable because that is the smallest infinite cardinality. Therefore, we have a number of obvious corollaries to this theorem.

[11.1.1] *The union of any denumerable family of denumerable classes is denumerable.*

[11.1.2] *The union of a countable family of countable classes that contains a denumerable class is denumerable.*

[11.1.3] *The union of a denumerable family of pairwise disjoint countable non-empty classes is denumerable.*

Theorem of countable union 11.2. *If X is an infinite class and Y is a countable class, then $union(X, Y)$ has the same cardinality as X.*

Proof. We have already seen that any infinite class has a denumerable subclass. Let $X = union(R, D)$ where D is denumerable and R is the relative complement of D in X, which means that $R = difference < X, D >$.

Lemma. The union of a denumerable class with a countable class is denumerable. This has already been demonstrated [11.1.2].

Therefore $union(Y, D)$ can be placed in a 1:1 relationship with D, and R is in a 1:1 relationship with itself.

Then $union(X, Y) = union(R, D, Y) = union(R, union(Y, D))$. It follows that $union(X, Y)$ has a 1:1 relationship with $union(R, D) = X$. □

Corollary 11.3. *If X is an infinite class and D is denumerable and $Z = difference<X, D>$ is infinite then $Z = difference<X, D>$ has the same cardinality as X.*

Proof. Z is a subclass of X. Therefore $card(Z) \leqq card(X)$. Because Z is infinite and D is denumerable, theorem 11.2 tells us that $card(Z) = card(union(Z, D))$. Because X is a subclass of $union(Z, D)$, $card(X) \leqq card(union(Z, D))$.

Therefore $card(X) \leqq card(Z) \leqq card(X)$. We conclude that $card(Z) = card(X)$. □

We have already seen that the class of all finite binary strings is denumerable. We can use this to prove the following important result.

Theorem of symbolic cardinality 11.4. *The class of all symbols of a non-empty alphabet is denumerable.*

Proof. The class of all symbols of a non-empty alphabet is infinite. Let x represent any one character of the alphabet. Then we have the sequence of symbols

$x, xx, xxx, - - -$ which has no last element. Now, we need merely show that the class of all symbols of the alphabet is dominated by the class of all finite binary strings.

Consider the class of segmented unary characters, SU. Each of these characters is uniquely represented by a binary string. If the alphabet is finite and of cardinality n, then we can reify its characters by the first n characters of SU. If the alphabet is infinite, then we can represent its characters by all of the characters of SU. In either case, every symbol is represented by a unique finite binary string. Therefore, the class of all symbols of this alphabet has a 1:1 relation to a subclass of the finite binary strings and is dominated by the finite binary strings. □

12. Order

The order type of a class or a set is a consequence of assigning order relations to pairs of distinct elements. This assignment is of particular importance for the classes or sets that are used in numerical computations and also for the understanding of ordinal numbers. There are several different but related ways of expressing an order relation. The simplest is trichotomous.

Suppose we have a class B and let x and y be elements of B. If B is totally and strictly ordered, then we can say for any elements x and y, that one and only one of these things is true: $x<y$, $y<x$, or $x=y$. The equality relation applies if and only if the elements are identical.

If B is not totally ordered then the trichotomy applies to some pairs of elements of B but not necessarily to all pairs of elements. Pairs of elements to which the trichotomy applies are said to be **connected** by the order relation. When we specify that a class is partially ordered, we include totally ordered classes in that category.

The $<$ order relation is defined to have the properties of being **irreflexive** ($x \not< x$), **asymmetric** (if $x<y$ then $y \not< x$) and **transitive** (if $x<y$ and $y<z$ then $x<z$). The equivalence relation, $=$, is reflexive, **symmetric** (if $x = y$ then $y = x$), and transitive. The $<$ relation is said to be a **strict order relation**.

When we speak of the order of any class of symbols that is generated by a simple finite recursion we shall normally mean the order of generation. When the trichotomous order of a class is total it is also said to be **linear**. This is the case for all infinite classes generated by simple finite recursion.

There is also a dichotomy that can be used. If two elements are connected then one and only one of these applies: $x<y$ or $y\leq x$, which is read as y **is less than or equal to** x. We call $\leq$ a **weak order relation**.

A very commonly used non-dichotomous definition of the order relation is this: If two elements are connected then $x\leq y$ or $y\leq x$ or both. When both are true then $x=y$. A nice property of this form is that only one order relation is employed. In this form, the order relation is reflexive, transitive, and **anti-symmetric** (if $x\leq y$ and $y\leq x$ then $x=y$).

With these definitions, we can use certain alternative forms: $x>y$ means the same thing as $y<x$, $x\geq y$ means the same thing as $y\leq x$ and, as usual, $x=y$ means the same thing as $y=x$. The natural numbers are linearly ordered, which means that the trichotomy applies to all pairs of natural numbers and that equality is true only in cases of numerical identity. The transitivity of $<$ assures that each symbol in any representation of the natural numbers can occur only once in the sequence and they are linearly ordered.

The ordering of the natural numbers is total and strict, in the sense that if the numbers x and y are not identical then either $x<y$ or $y<x$ but not both. This is not necessarily true of **symbols** representing natural numbers. For example, $1 + 1 = 2$, so the symbols that represent natural numbers are not strictly ordered. The use of $=$ with respect to the symbolic names of reified elements should be understood to refer to the elements themselves and not to the symbolic names. For example, if the universe of discourse is the natural numbers then x=y makes perfect sense but 2=3 is false because 2 and 3 are the names of distinct numbers.

Furthermore, every subclass of the natural numbers has a least element. This kind of order is called a **well ordering** which is special kind of linear ordering. When we recursively generate unique symbols for any particular representation of the natural numbers, the next symbol to be generated represents the least element of those that have not yet been generated. When we recursively generate unique symbols for any particular representation of the natural numbers, the first element that we encounter that is a member of a well defined subclass of the natural numbers is the least element of that subclass.

Every well ordering is a strict total ordering. This is so because every pair of elements that are not identical has a unique least element. Not every strict total ordering is a well ordering. We shall see that the rational numbers are an example of this.

When two classes have been placed in a 1:1 correspondence, it is possible to assign to one of them the ordering of the other. In that case they are said to be **order isomorphic**. Any infinite class of symbols that can be generated by simple finite

recursion will be order isomorphic with the natural numbers. All such classes are order isomorphic, using the order of generation. Two classes that are order isomorphic are also said to be of the same **order type**. It is easy to see that the positive integers and the natural numbers are of the same order type.

The order type of the natural numbers will be called NN ordering. This is an important special type of well ordering.

Consider X and Y to be two classes that are related by the bijection $f(x) = y$, where $x \in X$ and where $y \in Y$ is the image of x under f. Then Y is order isomorphic with X provided that $f(x_1) < f(x_2)$ if and only if $x_1 < x_2$ for any pair of elements in X.

When two classes have been placed in a 1:1 correspondence, it follows that their subclasses are also in a 1:1 correspondence. Whenever the two classes are order isomorphic their corresponding subclasses are also order isomorphic.

Every denumerable class can, by definition, be put in a 1:1 relationship with the natural numbers. Therefore, every denumerable class can be given NN ordering. Consequently, every denumerable class can be well ordered.

It is possible for denumerable classes to be given different order types and for one denumerable class to be given two different order types. An example of this is the class that is the denumerable union of disjoint denumerable classes of symbols. We have already seen that its elements can be represented by ordered pairs of natural numbers, such as $< j, k >$ which represents the kth element of the jth class. This is a class of symbols which is an infinite subclass of the NN ordered class of all symbols of a finite alphabet and it can be finitely recursively generated. Therefore. it can be NN ordered. Neverthless, if we order these elements lexicographically the 0th element of each of the original classes is not a successor element. $< 0.0 >$ is the foundation element and $< k, 0 >$ is a limit element for each natural number $k > 0$.

13. Ordering Classes of Symbols

We shall discuss universes (of discourse) of sets whose foundation classes are classes of symbols. In the recursive construction of these universes by successive classes, the names of these classes will be indexed by ordinal numbers. These ordinals will be a well ordered class of symbols. Therefore, it will be useful to discuss some ways of well ordering classes of symbols.

To begin with, we observe that any infinite class of symbols that we can generate by means of a simple finite recursive procedure has the order of the natural numbers (an NN order) which is the order of its generation. These classes form an equivalence family based upon order isomorphism, they are all NN ordered, and they all have the same cardinality, $\aleph_0$.

The class of all finite binary strings can be ordered as a member of this family. Consider the following generation.

The foundation class is just 0. The successor function, $nextbin(x)$ is algorithmic.

Let x represent any finite binary string. If x is not a string of all 1's, then $nextbin(x)$ returns $x+1$, using ordinary binary addition. However, if x is a string of all 1's then $nextbin(x)$ returns a string of all 0's which is 1 bit longer than x. In this fashion, every possible bit string of length k will be generated before starting on strings of length $k+1$. The universe of all finite binary strings has NN order as generated here and cardinality $\aleph_0$.

This is important because we have already seen that the class of all symbols of any non-empty finite alphabet can have a 1:1 relationship with some infinite subclass of the finite binary strings. Beause $\aleph_0$ is the smallest infinite cardinal, every infinite class of symbols must have cardinality $\geq \aleph_0$. Also, because we can NN order the finite binary strings, every denumerable subclass of an infinite class of symbols can inherit NN order, which is a special kind of well ordering, from the NN order of the finite binary strings.

Given any class of symbols of a finite or NN ordered alphabet, there are two commonly used ways of assigning a strict and total order to them. The more familiar of the two is usually called lexicographic. If the class is of finite cardinality, any sorting algorithm can order it lexicographically. We shall assume any alphabet to be finitely linear or else NN ordered.

All classes of symbols that are creatd by simple finite recursion are NN ordered by the order of generation and all NN ordered classes are well ordered. Now suppose we are presented with just an unordered class of symbols.

To lexicographically order any two finite strings of linerly ordered alphabetical characters, we begin with the leftmost character of each string. If these are not the same then the one with the lesser first character is the lesser of the two strings. If they are the same then we go one step to the right and compare the next pair of characters and so on until one of the following is true: either one of the two strings has ended in which case it is the lesser of the two or else the two strings have different characters

in some position, in which case the one with the lesser character in the first such position is the lesser of the two. If the two strings have the same character sequence and terminate at the same position then they are identical.

Every finite class of symbols has a lexicographic least element that that will be identified after a finite number of character-pair comparisons because the strings are all finite. The same will be true of any class of symbols having bounded length (that is, bounded numbers of characters per symbol). If a class of symbols is infinite then there may be no longest symbol, even though each symbol is of finite length.

Lexicographic ordering is not a well ordering for the class of all finite strings of an alphabet of two or more characters.

Proof. Let x be the least of the alphabetical characters and let y be the successor of x. Consider the subclass of finite strings each consisting of a string of all x characters followed by a single y. Now consider a string in that subclass consisting of k x's followed by a single y, where k is any positive integer. That string cannot be the least element of the subclass because a string of (k+1) x's followed by a single y is less according to lexicographic order. There is no least element of this subset. Therefore, lexicographic ordering is not a well ordering for this class of symbols. □

The second type of ordering will be called **canonical**. To find the least element of a class of symbols in canonical order, we first select the subclass of shortest length, by which we mean the subclass of symbols having the fewest alphabetical characters. This subclass will always exist because the length is a natural number and the natural numbers are well ordered. Then we select the least of these by means of lexicographic ordering. Scanning the class of alphabetical characters at each position from left to right we will always find a least character, assuming a well ordered alphabet. The symbols being compared are all of the same finite length, k, so the process will end at position k. At that point, there will be one or more least symbols remaining. If there are more than one, they must be identical, which cannot occur in a true class. Therefore,

The canonical ordering of a class of symbols is always a well ordering.

The order in which we generated the class of all finite binary strings by means of a simple finite recursion is the canonical ordering for that class. When the alphabet is finite and of $k > 1$ characters, the class of all finite strings can be similarly generated by identifying these characters with the numbers 0 through $k - 1$ and employing the add1 function in arithmetic base k as a successor function. When the current string is a sequence of all $(k - 1)$s its successor is a string of all 0s that is one bit longer.

On the other hand, when the alphabet is denumerable we can see that there are denumerably many limit points. In that case, the canonical ordering is a well ordering but it is not NN ordered.

The canonically ordered class of all symbols of a finite alphabet and the canonically ordered class of all symbols of a denumerable alphabet both have the same cardinality, $\aleph_0$, and they are both well ordered, but they do not have the same order type.

Furthermore, the class of all symbols of an NN ordered alphabet cannot be generated in canonical order by a finite recursion. Because finite alphabets are the only ones we know how to use, **we shall assume that our meta-language alphabet is finite.**

We now can justify the existence of $multicat(X, Y)$, where X and Y are both NN ordered classes of symbols. We do so by demonstrating that all elements of this class can be generated in a simple finite recursion, that is, each element will be reached after a finite recursion of algorithmic successor functions.

Because both arguments are NN ordered, the elements of the output will have a 1:1 relationship with the class of all ordered pairs of natural numbers. We can generate them, starting with the single element $< 0, 0 >$. This is the only pair that sums to 0 and it is the only pair in class C_0.

The next class, C_1, is the class of all pairs that sum to 1. The first element of this class is obtained by adding 1 to the right hand element of the only element of the previous class, giving us $< 0, 1 >$. There is one more element in $C_!$. This is obtained by subtracting 1 from the right hand element of $< 0, 1 >$ and adding 1 to the left hand element, giving us $< 1, 0 >$.

The general rule for constructing a class C_k is to begin with $< 0, k >$ and apply the following algorithm as long as it is applicable: going from right to left, subtract 1 from the right hand element and add1 to the left hand element. When the right hand element is 0, start the next class.

Because each class is finite, this is an example of finite recursion.

This procedure can be generalized to apply to any finite ordered family of arguments for *multicat*, each of which may be finite or NN ordered. The method is quite simple. Each class, C_k, has the sum of all of the numbers equal to k. Begin constructing the class with the lexically least element and then find the next least element and so on until the lexically greatest element that sums to k is reached. Then go on to construct C_{k+1}. Because each class is finite, each element will be reached within a finite number of algorithmic steps.

14. Ordinals and Ordinality

The natural numbers can be used as ordinal indices to reify the generated classes in a finite recursive class generator (e.g., C_0, C_1, C_2,...). This enables us to reason about the classes and to keep track of the order in which they were created and to determine which of any family of classes has the least ordinal.

Ordinals are well ordered classes, which means that every subclass of them has a least element. When we use them to index the classes in a recursive generator, there is always a next ordinal available. The next ordinal is the least one of those that have not yet been used.

In a finite recursion, the first limit class is the one corresponding to the union of all previous classes after a denumerable sequence of recursions. It is not a successor class but it contains all elements of the preceding classes. The customary symbol for this ordinal is ω (omega).

In the case of concatenation of symbols, a finite recursion is sufficient because the application of a concatenation function to any finite elements of the first limit class would not create any new elements. Specifically, when we have generated all finite strings by concatenation, the concatenation of any finite subclass of them would create a finite string that already exists. In other words, this universe is complete with respect to concatenation.

The generation of classes that contain sets will be different. In ZF theory, transfinite recursion is possible and it will be convenient to have transfinite ordinals to reason about it and to define the universe. Transfinite ordinals are essential to the reification of the classes of a transfinite recursion when the classes are not too numerous to be reified. For this reason we shall consider only classes of symbolic ordinals. We shall not need any others. (When we have limited our attention to symbolic sets we shall not need transfinite ordinals beyond $\omega + 1$.)

Transfinite **symbolic** ordinals exist. The least of them is conventionally represented by ω. This is the limit ordinal for any finite recursion. This symbol does not represent any natural number. It is the least ordinal that is greater than any natural number. It is the least transfinite ordinal. It is the limit ordinal of any NN ordered class.

The ordinal following ω is customarily represented by $\omega + 1$. If k is any natural number, then $\omega + k$ is the kth ordinal following ω. By this means, we have linearly ordered representations for all ordinals up to the second limit ordinal,
$\omega + \omega = \omega \cdot 2$. There exist symbolic representations for vastly larger ordinals but we shall not be needing them.

The ordinal number of recursions in a generative procedure can be limited by specifying a limit ordinal to be the least upper bound (lub) to the permitted ordinals.

When any ordinal α in the recursive generation of a class X of ordinals is specified, the ordered class of all elements with ordinals that are less than α is called the **initial segment** of X with respect to α. A convenient notation for this is $initseg < X, \alpha >$. When a limit ordinal is selected as the lub of an initial segment there is no greatest ordinal in the initial segment.

A successor ordinal can also be specified as an *lub*. For example, there is the ordinal that is usually called ω and the one that is its successor is $\omega + 1$. We can limit the recursion at this point by specifying that to be the least upper bound, *lub*. Then the initial segment is the ordered class of all ordinals less than $\omega + 1$.

When one ordinal class can be shown to be order isomorphic with a proper initial segment (i.e. one that is a proper subclass) of another it will be said to be of lower ordinality than the other.

Why do we consider limiting the ordinals in this fashion? Well, the ordinal *lub* can be a part of the definition of a universe of discourse. We are free to define the universe as we see fit. We may choose to do this wherever we see a likelihood that further recursion would not be useful. There may be no innate limit to a certain transfinite recursion but there may also be no good reason to pursue it beyond some interesting point.

Some interesting properties of well ordered classes can be deduced from the definition of a well ordering. It should be understood that a given infinite class of symbols may have more then one well defined order type.

Definition. *A class X is well ordered if and only if X is linearly ordered and every subclass of X contains a least element.*

When X is well ordered, the least element of any subclass of X is unique.

If X is a well ordered class then all subclasses of X are also well ordered.

An important result, a "proof" of which may be found in Halmos' section on Transfinite Recursion [1] is:

Comparability theorem for well ordered classes. *Given any two well ordered classes, they are either order isomorphic or one of them is order isomorphic to a proper initial segment of the other. Only one such order isomorphism is possible.*

A recursive construction for such an isomorphism of two well ordered classes of symbols will be found in the appendix. It should be read only after reading the body of this paper.

15. Characteristic Functions and Natural Number Ordering

According to the axiom of membership a particular element is either a member of a particular set (or class) or it is not a member. Suppose we have a finite set X and we wish to define a subset Y of it. One way to do this would be to list the members of that subset. This would define the extension of it. Another way would be to define a condition function, $condY(x)$, whose domain is all of the elements of X and whose range is the pair of values 1, 0. $condY(x)$ would return 1 when $x \in Y$and it would return 0 when $x \notin Y$.

For example, suppose we have the linearly ordered set $X =< a, b, c, d, e, f >$ in which we assume the order of the elements to be as written and we wish to define the subset $Y =< b, d, e >$. We could represent the condition for membership by the binary string 010110, which is a way of representing the $condY$ function for this subset. Each position in the binary string represents the value of $condY$ for the corresponding element of X. This representation of the subset depends upon having a finite linear ordering or a natural number ordering of the elements of X.

If a set is denumerably infinite it can inherit the order type of the natural numbers by establishing a 1:1 relationship with them and then any subset of it could be represented by an infinite binary string, that is, a string having a first element but no last element. This kind of representation for a subset is called its **characteristic function**. For example if we want to represent the subset of the positive integers that are divisible by three, then the characteristic function of $< 3, 6, 9, 12, \cdots >$ would be $001001001001\ldots$ Two common-sense difficulties are apparent. First, we can not actually print the entire infinite string and, second, Georg Cantor has proved that the class of all subsets of a denumerable set has a greater than denumerable cardinality. Therefore its elements are too numerous to all be computed or to be defined by any meta-language texts, or even to be reified by any collection of symbols. In ZF set theory, it is considered acceptable for undefinable elments to axiomatically exist. We shall consider restricting this freedom later.

If X is a set that has the order type of the natural numbers, then we can see that the class of all of its subsets has a 1:1 relationship with the class of all infinite binary strings, which represent their characteristic functions.

With respect to all subsets of any set having NN order, the class of these subsets can be given a linear ordering although it is not an NN ordering or even a well ordering.

Consider the infinite binary strings which represent their characteristic functions.

We shall define two infinite binary strings to be equal if and only if they are pairwise identical over all locations in the strings, which is to say if they are identical. If we know them to be different, then there must be one or more bit locations in which they differ. Furthermore, since the locations can be indexed by natural numbers (used as ordinals), those locations in which they differ must have a least member, say the kth, because the natural numbers are well ordered.

Then the subset which has a 0 at location k is defined to be the lesser of the two. This is the application of lexicographic ordering to semi-infinite binary strings. It is easy to show that this strict and total ordering has the transitive property.

Given any two infinite binary strings, there is no algorithm that can tell us, in a finite number of bit-pair comparisons, that they are identical. If they are different the comparison will stop in a finite number of steps but if that is false it will never stop. The proposition is semi-decidable by this means. Given a textual definition of two infinite binary strings, it may be possible to prove that they are identical or that they are different, but most infinite binary strings are not textually definable. They are too numerous.

16. Well Definedness

The scientific enterprise, as we now understand it, employs theory to construct satisfactory symbolic models of real phenomena. Although observation of nature is our guide, some useful compromises have been made to accommodate important mathematical needs. For example, the idea of infinity has been accepted in order to make the natural numbers complete with respect to addition, the signed integers have been introduced to achieve completeness with respect to subtraction, and the rational numbers have been created to achieve completeness with respect to division.

With the acceptance of denumerable infinities for mathematical completeness we are inclined to make similar compromises in our definition of well definedness but they will be carefully limited.

Let's begin with generability. We have seen that we can define finite recursive processes that can generate infinite sequences of such things as alphabetical characters

and all symbols of a **finite alphabet**. These things that are finitely generable are well defined. The unifying concept here is that each element of the generated sequence will be reached after a finite number of applications of an algorithmic successor function. In some cases the successor function may be complicated but is nevertheless algorithmic.

There will be some simple extensions to this strict condition. Suppose that we want to define subsequences of an NN ordered sequence. Every finite subsequence is well-defined because it is a symbol. What are we to do about defining the infinite ones?

In this case, characteristic functions provide a way. Every infinite subsequence has a unique characteristic function which may be represented as an infinite binary string and every infinite binary string represents a unique subsequence. There is a 1:1 relationship between the subsequences and their characteristic infinite binary strings. In particular, if the successive bits of a given infinite binary string are finitely generable, then we must consider the corresponding infinite subsequence to be well defined. Also, if the meta-language can provide a text that defines a subsequence then both the subsequence and its characteristic function are well defined.

We shall make use of this definition of *well defined* in defining a restriction of the real line to have only well defined elements.

17. Cantor's Ladder of Infinities

Before presenting the procedure we shall use for creating universes of sets, we offer a simplified example of an essential feature of ZF set theory. We employ a recursive creation of a universe of sets as follows.

Axiom of the empty set. *The empty set exists.*

The foundation for this procedure is the empty set.

Definition of the power set. *If X is a set, the power set of X, $PS(X)$, is the set of all subsets of X.*

Axiom of the power set. *If X is a set then the power set of X exists.*

If X is a set in our procedure then the successor of X is $PS(X)$.

Theorem of *wrap*. *If Z is a class of sets then wrap(Z) is a set.*

Suppose that the sets in our procedure are indexed by conventionally represented ordinals, so that the empty set is reified as X_0 and its successor as X_1, and so on.

This sequence has no end but we assume the completion of that procedure such that if any X_k is in it then its power set is also there. Therefore, nothing new can be created by applying the successor function to any already existing set.

Assume that we are at the first limit point. Its ordinal is ω. All of the existing sets are finite and indexed by finitely generable ordinals. Now, we can apply the theorem of *wrap* to the class of all existing sets. These are all finite but the cardinality of the resulting class is $\aleph_0$ and the cardinality of its wrap is also $\aleph_0$. This is the first infinite set in our construction. We can then apply the recursion once more, from ordinal ω to the next limit point, $\omega \cdot 2$. It will be sufficient to assume that $\omega \cdot 2$ is the *lub* of the ordinals for this exercise.

The succession of ordinals from ω on is $\omega+1, \omega+2, \omega+3, \ldots \omega \cdot 2$. When the recursion reaches the ordinal *lub* the procedure is complete. Now we want to examine the cardinalities of the infinite sets that have been created. Cantor's power set theorem is a necessary tool.

Cantor's power set theorem. *Every set is strictly dominated by its power set. If X is a set and $PS(X)$ is the set of all subsets of X then $X \prec PS(X)$. This is equivalent to saying $card(X) < card(PS(X))$.*

Proof. Observe that the elements of X have an obvious 1:1 correspondence with the singleton elements of $PS(X)$. Let's call this the function f such that $f(x) = x$ for all x in X. This is a function from X into $PS(X)$ but not onto, because there are no elements of X left to correspond to the remaining elements of $PS(X)$. $PS(X)$ dominates X.

We will now assume that there is another 1:1 function, g, from X onto $PS(X)$ and find that this assumption leads to an unavoidable contradiction. This will show that the assumption is false and that $PS(X)$ strictly dominates X.

Let Y be the unique set of all elements of X that are not contained in their associated elements, $g(x)$, of $PS(X)$. Being a subset of X, Y is an element of $PS(X)$.

Let z be the element of X that is associated with Y: $g(z) = Y$. Either $z \in Y$ or else $z \notin Y$. If $z \in Y$ then, by the definition of Y, it must not be an element of Y, a contradiction. Also, if $z \notin Y$ then, by the definition of Y, it must be an element of Y, also a contradiction. There is no alternative to the conclusion that the 1:1 relationship, g cannot exist. □

We observe in passing that this proof requires that the set $PS(X)$ exists. That requires the axiom of the power set. The axiom of the power set declares that the power set exists without the constructive requirement that all of its elements must

exist (if you want to construct a brick wall, you must have bricks). This irritant will be removed in the next section.

We already know that Cantor's power set theorem is true for finite sets. The number of subsets of a set X of finite cardinality k is precisely the number of their characteristic binary strings, which is 2^k and $2^k > k$ for all natural numbers, k.

It is interesting that Cantor's powerset theorem is also true for infinite sets. Not only is $card(PS(X)) > card(X)$, it is very much greater. Recall that the union of a denumerable class of disjoint denumerable classes is still only denumerable.

Let's return to our recursive generation process. The foundation set is the empty set, $\{\}$, which is of cardinality 0. The successor set is $PS(\{\})$. The symbol for this is obtained by wrapping a list of all subsets of the empty set in curly brackets but the empty set has only one subset, which is the empty set. Therefore, the power set of the empty set is $\{\{\}\}$, which is of cardinality 1. The successor of this set is $\{\{\{\}\}\{\}\}$, which has cardinality 2.

Expressions like this and, especially, longer ones are hard to read but we can simplify that task if we replace the open brace, {, by 1 and we replace the close brace, }, by 0. That results in 11100100 for the set of cardinality 2. Notice that the number of 1s and the number of 0s are equal. This will always be the case because the brackets occur in balanced pairs. We don't need a delimiter to identify the two subsets because a balancing algorithm can do that. It works as follows: proceeding from left to right and starting with zero we keep count by adding 1 for each 1 that we encounter and subtracting 1 for each zero. When the count goes back to zero we have identified a set. Removing the initial 1 and the terminal 0, we can then repeat the balancing procedure to identify the two subsets which are the elements of this set.

We can repeat the recursive application of the powerset creation endlessly. Each time we create a successor set with exponentially greater cardinality, but they are all of finite cardinality because if k is finite then 2^k is also finite. At the completion of this simple finite recursion we have created an infinite subclass of the finite binary strings.

Can we continue the generation process beyond this point? The answer is yes, according to the axioms. We take the *wrap* of the union of all of the finite sets that have already been created. This is a set of denumerable cardinality. Its power set a set having the cardinality, **c**, of the continuum. Then we can continue to apply the power set successor function endlessly. This will create a ladder of ever increasing infinite cardinalities.

There are two objections to this process from a scientific point of view. First, nature has not provided us with infinities that we must model. The creation of even denumerable infinities was a mathematical convenience in the interest of completeness. However, we shall see that the definition of the real line has a similar justification and it has trans-denumerable cardinality.

A second objection is more compelling. Objects in a scientific model are traditionally defined by meta-language texts. The cardinality of all texts of a non-empty countable alphabet is denumerable. Therefore, most of the elements of a trans-denumerable universe of discourse cannot be defined by texts.

There is no presently conceivable need for undefinable elements in a scientific model. If nature were to require such a model, it would force a drastic redefinition of the scientific enterprise. Absent such a need, most of the present curriculum of mathematical set theory can be (at least tentatively) regarded as scientifically inapplicable.

However, by simple modifications of the axioms, some of the ideas that have been developed for set theory will be useful in creating a symbolic set theory that avoids this problem. With such modifications, we can construct the **definate real line** which contains only the well defined elements of the real line.

18. Generating Classes of Pure Sets

A recursive definition by classes can be employed to constructively define a universe of sets in which the foundation class is empty, or else its elements are symbols, and all elements are either foundation elements or sets. We can again generate an infinite ladder of increasing infinite cardinalities but with axioms which will provide a clearer path to the restrictions we desire.

Definition. *A foundation class may be any well defined class of distinct symbols or the empty class.*

Axiom of existence. *A foundation class exists.*

Initially, we shall consider the case of an empty foundation class, that is, the case of pure sets in which there are no urelements (i.e., symbolic foundation elements that are not necessarily sets). It will be a simple matter to populate the foundation class afterward.

In our recursive definition of a universe we shall employ a class of ordinals: some specified, well ordered collection of symbols. Because they are symbols, the class of ordinals has a countable cardinality.

Axiom of ordinals. *A class of ordinals exists.*

Definition. *Any collection of elements in a given universe is a class.*

The ordinals are not elements of the universe being constructed. They are instrumental in the construction.

We also need a function to recursively create the successor classes and another function to create the limit classes of our universes.

Limit classes are created by the union function for classes.

Theorem of class unions. *The union of any family of classes of elements is a class of just those elements that are members of some class of the family.*

In addition to the union, we shall be using two functions whose domains are classes of elements. The first of these is *wrap*, which we have already defined and which converts a regular class (which will soon be defined) into a set that has the same extension. The difference between the class and the set is that a set is an element of the universe whereas a class is not an element. A class that is not an element is sometimes called a proper class.

Axiom of *wrap***.** *If X is a class then $wrap(X)$ is a set and an element if and only if X is regular.*

Every class in the universes we shall define here will be a proper class even if it is regular. By this means, we maintain a clear distinction between sets and classes. Regularity is the gatekeeper through which classes create sets.

The second function will be called *multiwrap*.

Definition of *multiwrap***.** *If X is a regular class then $multiwrap(X)$ is a class that contains the wrap of every regular subclass of X.*

The recursive function that creates successor classes called **cumulative classes** in our universes will be *multiwrap* and the successive classes will be reified and indexed by successive ordinals. If α is an ordinal and β is the successor ordinal of α and if CC_α is a cumulative class in the universe then the successor class is

$$CC_\beta = multiwrap(CC_\alpha).$$

We shall need a concept of rank, similar to its use in John von Neumann's cumulative hierarchy. Ranks are ordinal numbers. The ordinal of the foundation class and the rank of each member of the foundation class is the least ordinal, for example, 0.

Definition of rank. *The rank of an element is the ordinal of the least cumulative class of which it is a member.*

Definition of cumulative classes. *A cumulative class having ordinal α is a class that contains every element that has a rank that is less than or equal to α and no others.*

Let's look at the first few classes of the construction.

In constructing a pure set theory, the foundation class, CC_0, is the empty class, (), and the *multiwrap* of () is {}, the empty set.

$CC_0 = ()$
$CC_1 = \{\}$
$CC_2 = \{\{\}\}; \{\}$
$CC_3 = \{\{\{\}\}, \{\}\}; \{\{\{\}\}\}; \{\{\}\}; \{\}$

Here, we have used the semicolons to delimit the sets of the classes and we have used a comma to separate the subsets of the first set in CC_3. However, this notation rapidly becomes unreadable. If we substitute 1 for { and 0 for }, the first four classes are:
$CC_0 = ()$
$CC_1 = 10$
$CC_2 = 1100; 10$
$CC_3 = 11100, 100; 111000; 1100; 10$

We see that CC_3 contains all elements of rank 1 and rank 2 as well as all new elements of rank 3. The reason for this is that *multiwrap* reintroduces the empty set at each step. The recursive construction is such that no class can contain an element having a rank that exceeds the ordinal of that class.

The braces are introduced as balanced pairs and the delimiters are not necessary because we can separate the sets by means of a brace balancing algorithm.

If we perform a finite recursion of the pure sets, their union will be the class of all hereditarily finite pure sets. If we employ binary notation these are represented by finite binary strings, which are symbols. When canonically ordered, they will form a sparse but NN ordered subclass of the finite binary strings.

Cantor's theorem for multi-wrap. *If all subclasses of a class X are regular, then $X \prec multiwrap(X)$.*

Proof. If X is regular then $wrap(X)$ is the dominant set in $multiwrap(X)$, by which we mean that all of the other sets in $multiwrap(X)$ are the proper subsets of $wrap(X)$. The entire class dominated by $wrap(X)$ has an obvious 1:1 relationship with the elements of $PS(wrap(X))$. When all subclasses of X are regular we can see that $multiwrap(X)$ increases the cardinality of classes in the same way that $PS(X)$ increases the cardinality of sets. Furthermore, if the dominant set of a given class is X, then the dominant set of the successor class is $PS(X)$. □

Each set that is created by the recursion has the same rank as the ordinal of the first cumulative class in which it appears. While each cumulative class is indexed by a unique ordinal, many sets may share the same rank.

Definition. *By an* **ordinally bounded class***, we mean that the ranks of all of its members have an upper bound which is less than the least upper bound of the permitted ordinals.*

Definition. *A regular class is one that is ordinally bounded.*

This is not the ZF-axiom of regularity.

We are now able to present an axiom which, together with the choice of a foundation class and the permitted class of ordinals, will define the space.

Axiom of regularity 18.1. *A set is well founded and an element if and only if it is the wrap of a regular class.*

This definition of regularity has interesting and important consequences. If a class of elements satisfies the regularity condition, then all of its sub-classes also satisfy that condition. For that reason,

Theorem of regular subsets. *All of the subsets of a well founded set are well founded sets.*

If X is a regular class then the set $wrap(X)$ exists and all of the subclasses of class X are also regular. Therefore $multiwrap(X)$ also exists and it contains all of the subsets of set $wrap(X)$.

Every cumulative successor class will have an ordinal that is greater than the ranks of any of its previously created elements and is equal to the ranks of its newly created elements. Furthermore, there are always some new elements in any successor cumulative class because, according to Cantor's power set theorem, each successor class has a greater cardinality than its predecessor.

A **membership backtrace** is defined as follows. Given any non-empty set X, select any one of its elements. Lets call it BT_1. Every element of X must have

been created before X and it will have a strictly lower rank than X. Therefore, BT_1 cannot contain X. Now select any element of BT_1 and call it BT_2. The rank of BT_2 will be strictly less than the rank of BT_1. Therefore, BT_2 cannot contain either X or BT_1. Continue this process until either the empty set or an element in the foundation class is reached. One of these must be reached because the ordinals, being well ordered, have a least value and because the backtraces are indexed by strictly decreasing ordinals.

No new elements are created at any of the limit states. Therefore, every element of a backtrace has a successor ordinal as its rank.

Any entire chain of BT elements will be called a membership backtrace.

Because of the strict monotone decrease in rank of the sets along any membership backtrace from any set, there can be no loops in any of these backtraces. No set can contain itself nor can it be contained in any element in any of its backtraces.

This recursive construction cannot create a set of all sets. The regularity condition requires us to specify the ordinality of the space and it helps to define the universe. The construction we have specified conforms to that definition. Universes constructed in this manner are complete; every regular class has been wrapped.

This needs a little more explanation. If the ordinal lub is a succesor ordinal then the last class created will have that ordinal. However, if the ordinal lub is limit ordinal then that limit class will be regular because no new sets are created by the union function. All of its elements will be ordinally bounded. Therefore there is one last application of the successor function and the last class will be a successor class with ordinal $lub + 1$.

Suppose that the permitted ordinals have, as an upper bound, a limit class that is beyond the first limit class. The cardinality of the first limit class is denumerable and the cardinality of each class beyond that has a greater infinite cardinality than its predecessor. For the successor classes, this is proved by Cantor's theorem. Also, every limit class is of strictly greater cardinality than any of its predecessors.

Theorem. *The cardinality of a limit cumulative class, which contains the union of all of the preceding cumulative classes, is strictly greater than the cardinality of each of its predecessors.*

Proof. Let α represent a limit ordinal and let k represent any preceding ordinal. Because CC_k is a proper subclass of CC_α, its cardinality is less than or equal to that of CC_α. We shall show that it is strictly less.

Let ordinal k^+ be the immediate successor of k. We have the relation $CC_k \prec CC_{k^+} \preceq CC_\alpha$. There are two cases to be considered. In each case, it is easy to see by class dominance that $CC_k \prec CC_\alpha$. □

Therefore, if the ordinal upper bound is the second limit class, the cardinalities of the classes from the first limit class to the second limit class form an endless ladder of strictly increasing transdenumerable infinities. Then one last application of *multiwrap* follows. (No ordinal upper bound is specified in ZF theory.)

19. Universes Having Non-empty Foundations

Assume that we have a non-empty countable foundation class of symbols. In set theory these symbols are called **urelements** because they are not created from the empty set. They can represent whatever we choose or they can be just symbols. The construction of a universe with this foundation treats them just as symbols.

The cumulative classes will be indexed by ordinals. We employ well ordered, symbolic ordinals.

The foundation class, CC_0, may or may not be given some particular order. We shall assume a finite linear order at this time. We shall deal with infinite foundations later.

The constructive procedure is as follows. The classes constructed will be cumulative classes. Given any cumulative class of ordinal k, the class of its ordinal successor, k^+, is the union of two classes:

$$CC_{k^+} = union(multiwrap(CC_k), CC_0).$$

If CC_k is finite, then CC_0 must also be finite and CC_{k^+} is finite.

Let's look at the first few classes.

CC_0 is a non-empty finite foundation class of symbols. It's elements have rank 0.

CC_1 is the union of CC_0 and $multiwrap(CC_0)$, which contains the wrap of every subclass of the foundation class including the wrap of ( ). Taking the union of these gives us the cumulative class of all elements of rank 0 and of rank 1.

$CC_2 = union(multiwrap(CC_1), CC_0)$, which contains all elements of ranks 0, 1, and 2. The cumulative class of rank j will contain all elements of ranks 0 through j, which

is to say that the classes form a cumulative hierarchy. The names of the cumulative classes are indexed by their ordinals.

Cantor's theorem tells us that $multiwrap(CC_k)$, which is the class containing all elements of ranks 1 through k^+ and which contains a dominant set and all of its subsets, has a greater cardinality than CC_k. We have already seen that, when a class is infinite, taking the union of this class with, a countable class, does not change its cardinality. If $multiwrap(CC_k)$ is finite then so is CC_0, and the union will result in a greater finite cardinality.

The family of cumulative classes in this universe is indexed by their ordinals. Some of these are successor ordinals and some may be limit ordinals. The successor classes contain all elements of rank equal to or less than their own ordinals. The limit classes contain the union of all classes of lower ordinality and we have seen that their cardinalities are strictly greater than the cardinalities of their predecessors.

The recursive procedure creating a set theoretic universe can be transfinite because the *multiwrap* function can be applied even to those limit classes whose ordinals are less than or equal to the specified lub, and it will always create something new, specifically, elements of greater rank and a class of greater cardinality. While a limit class contains all elements of lower rank, the dominant set of its successor will be new and will contain all wrapped subclasses of the limit class as well as the foundation elements

Two interesting cases of such universes are the case of pure sets, which we have already visited, that is, a universe founded on the empty class, and the case of universes founded upon a denumerable class of urelements.

20. Universes Having Denumerable Foundations

In order to construct infinite sets, we shall need a transfinite recursion or an infinite foundation set. The powerset function, $PS(X)$, can only create finite sets from other finite sets because, for any set having finite cardinal k, its power set will have 2^k elements. Similarly, if we apply multiwrap to any class of k elements, we get a new class of 2^k elements.

Therefore, we now assume that we have a denumerable foundation class of symbols.

Cantor's theorem tells us that $multiwrap(CC_k)$, which is the class containing all elements of ranks 1 through k^+, has a greater cardinality than CC_k. We have already seen that, when a class is infinite, taking the union of this class with CC_0, a denumerable class, does not change its cardinality.

The family of cumulative classes in this universe is indexed by the specified ordinals. Some of these are successor ordinals and some are limit ordinals. The successor classes contain all elements of rank equal to or less than their current ordinal. The limit classes contain the union of all classes of lower ordinality. The recursive procedure can be transfinite because the *multiwrap* function can axiomatically be applied even to the limit classes (if they are regular) and it will always create something new, specifically, elements of greater rank in a class of greater cardinality.

What we know about the sequence of infinite cardinalities which we construct by this means is that they can be indexed by a class of symbolic ordinals. What symbols we use for naming these cardinalities is not particularly relevant, except to avoid confusion with other definitions.

Assuming a denumerable foundation class, the class indices can be given a least value of ω and a greatest value of the ordinal: lub or $lub + 1$. This would be consistent with transfinite constructions having a finite foundation class.

When the foundation class is denumerable and its ordinal is ω and the ordinal lub is $\omega \cdot 2$, the universe contains cumulative classes having an infinite ladder of Cantor's increasing infinite cardinalities. It does not matter what the foundation symbols are, so long as they are distinguishable from each other and from all subsequently created symbols.

21. Recursion

It will be convenient at this point to review what we have done with recursion.

The Peano axioms are the most commonly accepted tools for defining the natural numbers and for specifying their properties. However, we can do the same thing by recursively generating symbolic representations of the natural numbers. For this purpose we need only simple finite recursive generators. The advantage of this is that symbols represent the observables of scientific theory and the constructive use of symbols, rather than a list of axioms, facilitates comprehension.

The construction of a simple finite recursion requires the following:

1) a foundation, consisting of an initial element of the sequence to be generated (or a finite intial sequence),

2) a **successor** function, *suc* whose domain is the elements of the sequence and whose range is the elements of the sequence other than the initial element (or a finite initial sequence). Given any element, x, $suc(x)$ returns an element which has not been created earlier in the sequence, called the successor of x.
3) a process in which the successor function is applied to the last symbol (or symbols) reached, recursively without end, to generate the elements of an infinite sequence.

In order to have an endless recursion, the applicability of the successor function to each element must be hereditary and it must produce a new element at each step. When either of these conditions fails, the recursion must end or become repetitive.

The simplest example we have used (to generate a segmented sequence of symbols that represent the natural numbers) had a foundation of the one element, 0, and a successor function, $suc(x) = concat(x, 1)$. This produces the sequence 0, 01, 011, 0111 - - - . We also define the inequality, $<$, to mean prior location in this sequence. For example 01<0111 because 01 occurs before 0111 in the sequence.

We know that the following statements are true of this sequence.

The sequence has a least element, 0.
The sequence is infinite, meaning that it has no greatest element.
The sequence is strictly and totally ordered.
The sequence is well ordered.
The sequence is NN ordered.

Well ordered means that, given any subclass of the elements, there is a unique least element of them (which will be the first of them to be generated in the full sequence of elements).

We can also generate another sequence in which we start with 0 but use ordinary binary addition of 1 as the successor function. This will produce different finite binary strings to represent the natural numbers but this sequence will have the same listed properties as the first one, except that the symbols are not segmented. In fact, the two sequences are order isomorphic.

Order isomorphism is an equivalence relation and we can think of the natural numbers as being a concept that defines the equivalence family of such order isomorphic sequences. In creating mathematical structures we can use any convenient one of these symbolic sequences to represent the natural numbers.

Because any two simple finite recursive sequences can be placed in a 1:1 relationship they all have the same cardinality, which is designated by $\aleph_0$.

Because they are all order isomorphic, they all have the same order type as the natural numbers, which is sometimes designated by ω but which we shall call NN order.

Other interesting classes that have NN order can be generated by simple finite recursion. We have already seen how to generate the class of all finite binary strings by this means.

All simple finite recursive classes have the induction property with respect to the successor function. If the foundation element has property p and if the successor function is hereditary for p then every element of the class has property p. By hereditary, we mean that if any element x has property p then $suc(x)$ also has property p. By property, we mean any property that can be defined by a finite text in any sufficiently expressive meta-language such that we can unambiguously decide whether or not any element has the property.

Theorem. *The induction property holds for all simple finite recursive sequences because they are all well ordered and have only one limit point.*

Proof. We assume that the recursion is hereditary for property p and that the foundation element has property p. We also assume that there are some non-foundation elements for which the inductive property p fails and we deduce a contradiction.

Because the class of elements is well ordered, there is a least element x for which p fails. This means that the property holds for the predecessor of x, let's call it x^-.

This contradicts the assumption that the successor function is hereditary for p: that if p holds for x^- then it must also hold for x. □

The induction property can also apply to the classes of a finite class recursive generator. As an example, consider the generation of all symbols of a finite alphabet. The procedure is as follows.

The foundation class is the alphabet, AB. We shall reify the classes in the construction and index their names with some representation of the natural numbers: $C_0 = AB$.

Because the class names are indexed by the natural numbers as ordinals, they are a well ordered family. Therefore, if the successor function for classes is hereditary for a property p and if the alphabet has property p, then all classes in the construction will have property p.

In this case, the successor function is $suc(C_k) = multicat(C_k, AB)$.

One class property for which this generation is hereditary is finiteness (if the foundation alphabet is finite).

Although we have definitely reified names indexed by ordinals for these classes, it is noteworthy that the classes are actually functions of the ordinals. The functional relationship is as follows: given the ordinal k, we carry out a limited recursion up to the class, C_k, which is then the return value of the function.

When the successor function depends upon everything that has previously been recursively created, it is convenient to use cumulative classes. This is the case when we are creating a universe of sets, using the *multiwrap* function. When generating cumulative classes of sets, the successor function is:
$CC_{k+1} = union(multiwrap(CC_k), CC_k)$
or else
$CC_{k+1} = union(multiwrap(CC_k), CC_0)$
which happens to give the same result because $multiwrapCC_k)$ already contains all sets of ranks from 1 to $k + 1$. Only the elements of rank 0 need to be reintroduced by the union.

We can also define transfinite recursions. These are recursions that employ an ordinal lub beyond ω in conventional notation, that is, recursions that are ordinally not finite. The ordinals of such recursions have limit ordinals at intervals of ω. At each of these limit ordinals the class is not a successor class: it is a limit class which contains all previously reached elements but contains no new elements. It is created as the union of all previous classes. A transfinite recursion is possible when the successor function is applicable at the limit classes and beyond and it creates some new elements when it is applied. If the hereditary properties of the classes remain hereditary at and beyond each limit point then the induction property is valid for the classes.

When any class is cardinally transdenumerable, its elements cannot all be well defined. The ones that can be well defined by a text in the meta-language are, at most, denumerable.

22. Integers and Rational Numbers

Without reviewing the well known facts of arithmetic, we observe that the natural numbers are complete with respect to addition but they are not complete with respect to subtraction. In particular, if x and y are natural numbers and $x < y$ then $x - y$ is undefined. This condition is remedied by the definition of the signed integers, which provide the completion of the natural numbers with respect to subtraction.

The signed integers can be represented by the union of the positive integers, the negative integers and 0. The individual symbols can consist of a single sign bit left concatenated to a binary string representing the magnitude. In this case, $+0 = -0 = 0$. The signed integers include the natural numbers as non-negative integers.

Like the natural numbers, every integer has a unique successor. However, in the domain of arithmetic, each integer including 0 also has a unique precursor. These properties permit us to say that all of the signed integers are discrete both above and below. There is no least integer so the integers are arithmetically not well ordered. They can be well ordered by letting each negative integer immediately follow its positive counterpart but this is not arithmetic order.

In arithmetic order, the signed integers are totally strictly ordered so they are linearly ordered. They have no least element so they are not well ordered. Nevertheless, the class of all signed integers greater than any specified integer is well ordered and the class of integers less than some specified integer is such that all of its subclasses have a greatest element.

Consider the class of equations: $a \cdot x = b$, where a is a positive integer and b is a natural number. Sometimes the solution, x =b/a, is a natural number but frequently it is not. The non-negative rational numbers represent the completion of the natural numbers with respect to division by a positive integer and they include the natural numbers because b=b/1 for all natural numbers, b.

Suppose we wish to order the two non-negative rational numbers, a/b and c/d. Let rel represent any one of the relations: $<, \leq, >, \geq, =$. We say that $(a/b)rel(c/d)$ if and only if $(a \cdot d)rel(c \cdot b)$. Notice that the last relation is a comparison of two natural numbers and we have multiplied both members of the original relation by the same positive integer, $b \cdot d$. The order of the non-negative rational numbers is complete and therefore linear.

It is clear that the ratio of two integers, a/b, being a finite string of characters, is a symbol. Therefore, the class of all rational numbers is denumerable (i.e., of cardinality $\aleph_0$). There is, however, a considerable redundancy in this representation because $n \cdot a/n \cdot b = a/b.$, where n is any positive integer. The redundancy can be eliminated if we limit the representation of any rational number to a numerator and a denominator that are relatively prime by canceling out any common factors. What remains includes the positive integers and is therefore still infinite and it has cardinality $\aleph_0$.

It is not difficult to extend this discussion to include the signed rational numbers. This can be done with a single sign bit because if a and b are positive integers then $-a/b = a/-b = -(a/b)$. Therefore, the following collection of ordered triples can

represent all rational numbers other than 0: $< sgn, a, b >$ where sgn represents the sign bit. For example, we can use the code 1 for + and 0 for -. If we restrict a and b to be relatively prime positive integers then the union of this class of triples and 0 can be in a 1:1 relationship with the class of rational numbers.

Furthermore, if we use the segmented unary codes for a and for b then we can concatenate the elements of each triple without ambiguity and we get the family, $multicat< sgn, a, b >$, which is 1:1 with the rational numbers. In this case, a and b are restricted to relatively prime pairs of positive integers and we take the union of these triples with 0. These elements are all finite binary strings and therefore this class is denumerable.

There is another simple representation for the signed rational numbers which is also useful for the real numbers (which we shall define in the next section). In this representation, $x = k + y$, where x is a signed rational and y is a non-negative fraction. Here, k is the greatest signed integer that is less than or equal to x while $0 \leq y < 1$. This interval of values for y is also called the semi-closed unit interval, [0, 1).

Given any two rational numbers in this form, the one with the smallest integer part is the smaller of the two. When the integer parts are equal the one with the smallest fractional part is the smallest.

The rational numbers have two interesting properties. The first one is that they are arithmetically linearly ordered. Any two rational numbers which are not identical have a unique least element. Nevertheless, they are not arithmetically well ordered because, like the signed integers, this class has no least element.

The second property is that, unlike the integers, they are dense. A point x is dense if, given any other nearby point y, there is always another point that is between x and y. From this, we can see that if x is a dense point then there are infinitely many other points in any neighborhood of x. In the case of the rational numbers they are all dense (both above and below) because, given x and y, there is always $(x + y)/2$ between them. Therefore, the subclass of rational numbers in any open interval, for example $0 < x < 1$, has no least element and even the positive rationals are not arithmetically well ordered.

There are, however, well ordered classes (of ordinals) which have semi-dense limit points. The limit points in any transfinite class of ordinals are dense below but discrete above. They each have a unique successor.

23. Real Numbers

It is known that certain infinite sequences of rational numbers converge to a limit and that some of the limits are rational numbers but some are not rational [wikipedia irrational numbers].

Familiar examples of irrational limits are π and $\sqrt{2}$. The real numbers, which include the rational numbers, are the completion of the rational numbers with respect to convergent infinite sequences. The class of real numbers is sometimes called the continuum or the real line and the individual real numbers are sometimes called points on the real line.

We shall use the density and the linear order of the rational numbers as a framework upon which we shall construct representations of the real numbers as the limits of convergent rational sequences.

The class of signed rational numbers includes the signed integers. We shall assume that the class of real numbers includes the signed rational numbers and certain other numbers, called irrational numbers. We also assume that the real numbers are linearly ordered, as are the rational numbers. As we shall define them the real numbers are assumed to have the following properties.

1) the class of real numbers includes the signed rational numbers.
2) If x is a real number there exists a signed integer which is greater than x and there is a signed integer which is less than or equal to x. (This assumption is quite reasonable because the signed integers are doubly infinite.)
3) the real numbers are linearly ordered.

We interpret the last of these properties to imply the following dichotomy. There exists a function, $ordr< x, r >$, where the domain of x is the real numbers and the domain of r is the signed rational numbers, which returns 0 whenever $x < r$ and which returns 1 whenever $x \geq r$.

What we need now is to find a suitable representation for he real numbers and to study their properties.

For any real x, there exist integers that are greater than x. This class of integers has a lower bound so there is a least integer which is greater than x, which we shall call least integer upper bound of x, $liub(x)$, and there is a greatest integer, $liub(x) - 1$, which is less than or equal to x, which we shall call greatest integer lower bound of x, $gilb(x)$. We know that x lies in the semi-closed interval, $[gilb(x), liub(x))$, of unit length. These bounds are easily found by either a finite linear search procedure or

a finite binary search procedure. Both of these procedures are algorithmic and they will terminate with an answer.

We shall now seek to generate a representation for x by means of a finite recursive binary search procedure. We acknowledge at the start that the the *ordr* function may or may not be computable and we shall revisit this question after we have found a representation for the real numbers.

First initialize glb_0 to be a representation of the signed integer $gilb(x)$. Then $x = glb_0 + y$ where y is in the semi-closed unit interval, $[x - glb_0, x - glb_0 + 1)$. We shall now seek a representation of the non-negative real fraction $y = x - glb_0$.

Initialize lub_0 to be $glb_0 + 1$ and $mp_0 = (glb_0 + lub_0)/2 = glb_0 + 1/2$ is the mid-point of the initial interval.

Now apply the following procedure recursively without end. There are two possibilities.

If $ordr< y, mp_k >= 0$ then y is less than the k'th midpoint. In that case we set $b_{k+1} = 0$ and $glb_{k+1} = glb_k$, $lub_{k+1} = mp_k$,
$mp_{k+1} = (glb_{k+1} + lub_{k+1})/2$. We have not increased the value of the glb but we have lowered the lub to the value of the previous midpoint, thereby decreasing the length of the interval by half. These actions are associated with the value, $b_{k+1} = 0$.

If $ordr< y, mp_k >= 1$ then y is greater than or equal to the k'th midpoint. In that case we set $b_{k+1} = 1$ and $glb_{k+1} = mp_k$, $lub_{k+1} = lub_k$. As before, $mp_{k+1} = (glb_{k+1} + lub_{k+1})/2$. We have increased glb to the value of the previous midpoint, thereby decreasing the length of the interval by half. These actions are associated with the value, $b_{k+1} = 1$.

Each time that $b_k = 1$ we add 2^{-k} to the lower bound value of the fraction y, which is the new value of glb.

At each step $glb_k \leq y < lub_k$. We are generating a sequence of nested semi-closed intervals each of which includes the glb but not the lub and such that their lengths converge by successive multiples of 1/2 toward 0. Consequently, the sequence of glb's converges to y.

Our representation for y is $y = \sum_{k=1}^{\infty} b_k \cdot 2^{-k}$.

Consider the NN ordered class, $B =< 2^{-1}, 2^{-2}, 2^{-3}, - - - >$. The sum of these elements is $\sum B = 1$, which can be demonstrated as follows, using term by term cancelations. $\sum B - (\sum B)/2 = 1/2$, therefore $(\sum B)/2 = 1/2$ and $\sum B = 1$.

Because $y < 1$ and each b_k is either 0 or 1, it is clear that y is represented by the sum of a proper subclass of B. Furthermore, the NN ordered binary string b of all of the b_k is precisely the characteristic function of that subclass.

One more interesting fact about b is that, if you precede it by a binary point, it is the commonly used representation of a fraction in binary arithmetic.

In the real world of scientific computation, we do not generate the full infinite representation of y. Instead, we truncate it and compute the sum of the first n terms, which is called the nth partial sum and which is a rational number. Let us therefore examine how accurate such an approximation can be.

The first n terms of the infinite series will be called the nth **prefix** of y and the remaining terms will be called the nth **tail** of it. The sum of the nth prefix will be called the nth **partial sum** and the sum of the nth tail will be called the nth **residue**.

Let P_n^+ represent the nth partial sum of y and let T_n^+ represent the nth residue of y.

$y = P_n^+ + T_n^+$, so $P_n^+ = y - T_n^+$ and the truncation error is $-T_n^+$. We can easily put an upper bound on the truncation error, because
$T_n^+ = \sum_{n+1}^{\infty} b_k \cdot 2^{-k} \leq \sum_{n+1}^{\infty} 2^{-k} = 2^{-n} \cdot \sum_1^{\infty} 2^{-k} = 2^{-n}$. The magnitude of the truncation error of the nth partial sum decreases exponentially to 0 with increasing n.

The rational numbers are everywhere dense so any well defined real number can be approximated as well as desired by a rational number. Whenever a real number has been adequately approximated by a rational number by any means, it is possible to compute an approximation in binary format by using the binary search procedure. In this case, the function $ordr<y, r>$ is computable because it requires only the comparison of two rational numbers. If desired, the upper bound of the rational approximation may also be put into binary format.

Binary fractions that have a tail of all 1s are redundant because $\sum_{j+1}^{\infty} 2^{-k} = 2^{-j}$ and they are not created by the binary search procedure. The equivalent forms that have a tail of all 0s are generated instead. Therefore, there is no last 0 in the representation of any of the fractions. They all have an infinite number of 0s.

We shall call the class of infinite binary strings that have no last 0 the **canonical binary strings**. We shall now show that the class of all infinite binary strings has a greater cardinality than the natural numbers and that the class of canonical binary strings has the same cardinality as the class of all infinite binary strings. This

cardinality is called the cardinality of the **continuum** and is sometimes denoted by **c**.

Theorem. *The class of all infinite binary strings has a cardinality which is greater than the cardinality of the natural numbers.*

Proof. We shall assume that the class of all infinite binary strings is denumerable and then derive a contradiction. This proof is essentially Cantor's diagonal proof.

If the infinite binary strings are denumerable then they can be put into an order isomorphism with the natural numbers. Therefore, each such string can be uniquely identified with a natural number. Also, the individual bits in each string may each be identified with a natural number according to their positions in the string. It follows that each bit in the class of all infinite binary strings can be associated 1:1 with an ordered pair, such as $< j, k >$, which represents the kth bit of the jth string.

Now, we can show that there is another infinite bit string which has been omitted from this class. This new bit string is created as follows. Let's call the bits for which $j = k$ the diagonal bits. We complement each diagonal bit, changing a 0 into a 1 and changing a 1 into a 0. The infinite string of all of these complemented bits cannot have been in the assumed class because it differs in 1 bit from each of the strings. This contradicts our assumption that the entire class of infinite binary strings is denumerable.

However, the class of all infinite binary strings dominates the class of all finite binary strings. For example, change every finite binary string into an infinite one by left concatenating it to an infinite string of all 0s . The result is a subclass of infinite binary strings that is 1:1 with the finite binary strings. Therefore, since we now know that their cardinalities are not the same, we know that the class of infinite binary strings has a trans-denumerable cardinality. □

The cardinality of the infinite binary strings is represented by **c** and $\mathbf{c} > \aleph_0$.

Theorem. *The cardinality of the canonical binary strings is* **c**.

Proof. The canonical bit strings are a subset of the infinite bit strings and are therefore dominated by them. We can also show that they dominate the infinite bit strings as follows. In each of the infinite bit strings, insert a 0 between every pair of adjacent bits. The resulting bit strings have an obvious 1:1 relationship with all of the infinite bit strings but they are also a proper subset of the family of canonical bit strings. □

Because the canonical binary strings are 1:1 with the non-negative real fractions, we are presented with a problem. Each well defined element must be definable by some text in the meta-language. The cardinality of such texts is $\aleph_0$. Most of the real fractions are undefinable! [11.3] We must contemplate the linear ordering of a universe of real numbers, most of which are undefinable. This is an embarrassment from a scientific point of view which we shall address later.

Summary 23.1. If x is a real number it may be represented in the form, $x = glb + y$, where glb represents the greatest signed integer that is less than or equal to x and y represents a real fraction within the unit semiclosed interval, $[0, 1)$. We have seen that every real fraction can be represented as the ordered sum of the elements of some proper subsequence of the ordered class, $B =< 2^{-1}, 2^{-2}, 2^{-3}, - - - >$. B is an NN ordered class which sums to 1 and whose elements rapidly converge to 0. Every proper subsequence of B sums to a real fraction. The characteristic functions of these subsequences are infinite binary strings. The binary strings that have a tail of all 0's represent the finite subsequences. The binary strings that have a tail of all 1's are redundant because each of them is numerically equal to the sum of a finite subsequence. The class of subsequences which omits these redundant ones will be called the canonical class and their characteristic infinite binary strings will be called the canonical binary strings. When a canonical binary string is preceded by a binary point, it presents the usual notation for a binary fraction. Arbitrarily good rational approximations to the real fractions can be obtained by truncating the canonical strings at sufficiently many bits. The cardinality of the canonical binary strings is $\mathbf{c}$, so most of thm cannot be well defined.

24. More properties of the Real numbers

Theorem. *The canonical binary strings are arithmetically linearly ordered.*

Proof. If two of the canonical binary strings are distinct then there must be a location where the two have different values. The locations in these strings are NN-ordered so there must be a least location in which they differ. The string which has a 0 at this location is easily seen to be the characteristic function of the subset of the class B that has the lesser sum. It represents the smaller binary fraction because it cannot have a tail of all 1s. □

This ordering of the canonical binary strings is lexicographic. This is not a well ordering, which is not surprising because the canonical binary strings are dense.

Corollary. *Within the class of canonical representations of the real fractions, lexicographic ordering is the same as numerical ordering.*

This is not true of the class of all infinite binary strings. A lexicographic ordering is not the same as a numerical ordering because of the redundancy of the strings that have tails of all 1s. Although they are numerically equal, the redundant form is lexicographically less.

Theorem. *The cardinality of the canonical real fractions between any two distinct real fractions is* **c**.

Proof. Since the fractions are distinct, their corresponding characteristic functions must differ and there must be a first position in which they differ. The smaller of the two will have a 0 in that position. Furthermore, it will have an infinite number of zeros beyond that position. If we replace the zeros in those infinitely many subsequent zero positions by the binary digits from any canonical bit string that is not all 0s, we get a new binary fraction which is greater than the smaller original but not as great as the larger of the originals. The cardinality of these new strings is **c**. □

While the rational fractions are denumerably dense, the real fractions are trans-denumerably dense.

Theorem. *The class of all non-negative real numbers has cardinality* **c**.

Proof. The non-negative real numbers can be represented by the sums of ordered pairs $< n, k >$ of which the first element is a natural number and the second element is a non-negative real fraction. The natural numbers are 1:1 with a denumerable subset of the finite binary strings and the real fractions are 1:1 with the canonical infinite binary strings. Clearly, if we fix the natural number part as 0, we get a subset of the non-negative real numbers, so the non-negative reals dominate the real fractions.

Now let's employ a unary segmented representation of the natural numbers as strings of 1s each terminated by a single zero. We can left concatenate such a natural number representation to the infinite binary string representing the real fraction. When we do this, the first 0 in the resulting string unambiguously separates the natural number from the fraction. The result is a canonical infinite binary string and the real numbers so represented are a subset of the canonical binary strings. Therefore, the real fractions dominate the non-negative real numbers. □

Theorem. *The class of all real numbers has cardinality* **c**.

Proof. The negative real numbers have an obvious 1:1 relationship with the positive real numbers and therefore also have cardinality c. Lets place a sign bit (1 for + and 0 for -) in the initial position of the infinite binary strings that represent real numbers to distinguish the negative from the non-negative strings.

The non-negative real numbers are a proper subset of the real numbers. Clearly, the real numbers dominate the non-negative real numbers.

Also, any real number can be uniquely represented by a canonical infinite binary string. Therefore, the real numbers are dominated by the canonical infinite binary strings. □

Theorem. *The cardinality of any finite dimensional real space is* **c.**

Proof. Let's consider the two dimensional case. The elements of this space are in a 1:1 relationship with ordered pairs of canonical infinite binary strings having cardinality **c**. Therefore, its cardinality is at least **c**. If we bitwise interleave each such pair of strings we get a new canonical infinite binary string, so the cardinality of the two dimensional space is no greater than **c**. We can also bitwise interleave any ordered finite ntuple of canonical binary strings, so the same method establishes the result for finite dimensional spaces. □

25. Shrinking the Universes

We have constrained this presentation of set theory to sets that are symbolically founded and which have symbolic class ordinals. Nevertheless, we are faced with an embarrassment of riches. All cumulative classes of sets having cardinalities larger than $\aleph_0$ have transdenumerably more elements than we can define by any meta-language texts and there are denumerably many such classes of sets having an infinite ladder of greater cardinalities, even in a universe generated by a finite foundation and trans-denumerable recursion up to ordinal $\omega \cdot 2$.

When we have constructed a denumerable class of symbols, even one application of the *multiwrap* function will give us a class of sets which has cardinality **c**, having trans-denumerably more elements than we can well define. We have already seen this in the definition of the real line.

How large a universe do we really need? Science is an open ended enterprise and we cannot offer a definitive answer to that question. What we do know is that the continuum is a very useful mathematical concept and there is not now any scientific demand for greater cardinalities. However, even the continuum is a boundary region

in which most elements are not definable but in which many scientifically important irrational numbers are well defined.

What might be more desirable for scientific purposes is a universe containing only all of the elements that can be well defined. Such a universe could have only denumerable cardinality but it would contain every element that science could define, including irrational numbers. This can be achieved by modifying the axioms.

26. Symbolic Sets

Symbolic sets universes contain only symbolic ur-elements and well defined sets whose elements are all symbols.

We shall now modify the definitions related to the *wrap* and *multiwrap* functions so as to define universes of sets such that all sets are well defined and the elements of all sets are symbols.

The first step is to redefine regularity without explicit use of the concept of rank.

A class is ss-regular if and only if it is well defined and all of its elements are symbols.

This is a significant restriction. All finite classes of symbols are regular but most infinite subclasses of an NN ordered class of symbols are not regular because they are not well defined. The well defined infinite subclasses are important but they are not symbols. Therefore, no class that contains the ss-wrap of one of them as an element is regular. This will limit the recursion.

The *ss-wrap* of a class exists and is a set if and only if the class is ss-regular.

It should be clear that *multiwrap*, applied to finite classes is not the same function as *multiwrap* applied to regular denumerable classes. For example, when applied to a finite class, it returns a larger finite class which is well defined and regular and it can be algorithmically constructed. None of these things are true when it is applied to a denumerable class. We need a new function when applying it to a denumerable class and it is sensible to change its name.

We want to restrict *multiwrap* so as to avoid creating undefinable sets.

The *ss-multiwrap* of an ss-regular class, X, is a class whose only members are the wraps of all of the well defined subclasses of X.

We have now restricted both the domain and the range of *ss-multiwrap*. Nevertheless, the generative process will proceed from a finite symbolic foundation class as before until the denumerable class is created at ordinal ω. It is at this point that *ss-multiwrap* parts company with *multiwrap*.

If a class, X, is regular then all of its well defined subclasses are regular. This is true because the elements of X are all symbols. If X is infinite then some of its well defined subclasses are infinite and therefore not symbols. Because of this, *ss-multiwrap*(X) is not regular and no further application of *ss-multiwrap* is possible.

If we start the generation of a symbolic sets universe with an empty or finite foundation class then the generation will procede as before up to the first limit point. The class at CC_ω will contain all of the hereditarily finite sets and it will be ss-regular. Applying *ss-multiwrap*, the successor class at ordinal $(\omega+1)$ will not be regular and the recursion must terminate at that point. The universe will contain all sets that have been created, including the finite sets and the well defined infinite sets.

We have said nothing about the meaningfulness of the sets. That is a serious philosophic and linguistic problem and it has much to do with the application. The definition of the meta-language, other than having a finite alphabet, and the determination of meaingfulness are questions far beyond the scope of this paper. However, it is very easy to apply the restrictions of symbolic sets to the definition of the definate real line.

27. Well Defined Symbolically Founded Universes

The property of symbols and classes of symbols of being well defined is itself in need of an expanded definition.

We already know that all symbols of a linearly ordered finite alphabet are finitely generable and can be NN ordered. All elements of such a class and the entire class are considered well defined. This assumption about the entire class is not trivial. While all elements are finitely generable, the entire class is not. We must assume that the generation of this class can be carried out to completion. We accept all classes of symbols whose elements can be finitely generated as well defined classes. As a consequence of this mode of creation, all well defined denumerable classes of symbols can be NN ordered. We may ignore this property at times but we lose no generality when we make use of it.

What we must now do is to consider which denumerable subclasses of NN ordered symbols are well defined.

The criterion we used earlier is that a subclass Y of a well defined class X is well defined if we can define a condY(x) algorithmic function such that condY(x)=1 if x is an element of the subclass and condY(x)=0 otherwise. The definition of the algorithm must be a finite text in some suitably expressive meta-language and the algorithm must terminate with an answer.

Whenever NN order for the entire class of symbols has been established, there exists a well defined 1:1 relationship between the subclasses and their characteristic binary strings, each of which is representable as an infinite binary string. Therefore, if a subclass is well defined, its characteristic function can be represented as a well defined infinite binary string. Furthermore, every well defined infinite binary string represents a well defined subclass. The well defined subclasses and their characteristic functions are 1:1.

The advantage of using well defined infinite binary strings is that, once the symbols have been given natural number order, these strings are independent of the particular symbols being used. **Every generable infinite binary string is the characteristic function of a well defined subclass of an NN ordered class.**

Because the class of all symbols is denumerable, every infinite subclass of them is denumerable. Furthermore, there is no doubt that we can define infinitely many infinite binary strings. As a simple example, consider the generation of binary representations of all of the positive rational fractions. Each bit of these is finitely generable by means of a division procedure in binary arithmetic.

Suppose that an NN ordered foundation class is well defined and all of its elements are symbols. The family of its subclasses has cardinality **c** and only a denumerable cardinality of them can be well defined. If we denote the ordinal of the foundation class by ω then the new class we can create will have an ordinal that is conventionally represented by $\omega+1$. We accept this new class and its elements as well defined.

Some of the new sets created by this application of *ss-multiwrap* will contain infinitely many symbols and have rank $\omega + 1$. These sets are not symbols. Becuse we have defined ss-regularity to require that the class must have only symbolic elements, that would limit *ss-multiwrap* to just one application to any NN ordered class of symbols.

If the foundation class was generated by our recursive set generator and contains all of the finitely hereditary sets, then all of the the finite sets at ordinal $\omega + 1$ have been created before and have lower ordinals.

If, on the other hand, the foundation class is just any NN ordered class of symbols, then a single application of *ss-multiwrap* will create all of the finite sets as well as

the well defined infinite subsets. The finite sets will be those whose characteristic binary strings have a tail of all 0's.

28. The Definate Real Line

We have already seen [23.1] that the arithmetically NN ordered class $B =< 2^{-1}, 2^{-2}, 2^{-3}, \cdots >$ is a sequence of elements that converge to 0 and its sequence of partial sums converges to 1. B is easily seen to be a finitely generable sequence and is therefore well defined. To generate it the foundation element is 2^{-1} and the successor function is multiplication by 2^{-1}.

Because class B has only symbolic elements and is well defined we can apply *ss-multiwrap* to it. The resulting class, *ss-multiwrap*(B) contains the *ss-wrap*s of all of the well defined subclasses of B. Infinitely many of these subclasses of B are infinite and therefore not symbols so no recursion is possible beyond this point.

The well defined subclasses of B are 1:1 with their characteristic functions and these are all of the generable infinite binary strings. This gives us a neat way to characterize the entire class. However, it will not be necessary to compute all of them.

As we have seen before, the generable characteristic binary strings that have a tail of all 1's are redundant, being numerically equal to other strings that have a tail of all 0's. The ones that have no last 0, the generable canonical infinite strings, are 1:1 with the definate real fractions. This tells us that the cardinality of the definate real fractions is $\aleph_0$. We have also seen that these characteristic binary strings, when preceded by a binary point, are the actual representations of fractions in binary arithmetic.

Furthermore, the entire definate real line is denumerable and definate real finite dimensional spaces are also denumerable.

The important thing about the definate real numbers is this: **if and only if it can be defined, then it is there.** From the point of view of scientific computations, shedding the undefinable elements seems to have cost us nothing.

When we defined the real numbers [23.0], we employed a function, $ordr< x, r >$, for ordering the real numbers with respect to the rationals in the execution of a binary search procedure. The x variable represents a real number and r represents a rational number. We delayed a discussion of the computability of this function because we did not yet have representaions for the real numbers . It is now possible to deal with that question.

The rational numbers are a subclass of the reals. If we want to find a representation of a non-negative rational fraction, given to us as a ratio of a natural number to a positive integer, each comparison during the procedure is merely a comparison of two rational numbers and is easily computable. The procedure becomes a form of division in binary arithmetic. One other class of representations is also trivially computable. We know that the representations of the non-negative binary fractions as well defined infinite canonical binary strings must be fixed points for this search procedure because changing even one digit would change the arithmetic value of the result. It cannot be compensated by any changes in the tail. Because lexicographic ordering is equivalent to arithmetic ordering in a population of definate canonical infinite binary strings, these computations are performed by a very simple algorithm. Inasmuch as we cannot perform computations upon undefinable strings, we are well rid of that question.

It is quite clear that the well defined strings are arithmetically dense because they include the rational fractions. Their truncations can approximate any real fraction to any desired accuracy.

Any canonical bit string that can be defined by a text in a meta-language with a finite alphabet is the characteristic function of a definate non-negative real fraction. When preceded by a binary point, it is the binary arithmetic value of that fraction.

These observations suggest that, for the foreseeable future, binary computations with finite binary strings will be sufficient for scientific calculations.

29. References

1. Paul R. Halmos, Naive Set Theory, Springer, 1974
2. Derek Goldrei, Classic Set Theory, Chapman & Hall/CRC, 1996

30. Appendix

Suppose that X and Y are two well ordered classes of symbols. Having symbolic elements, both classes will have countable cardinality but they may be ordinately unequal. We shall construct an order isomorphism beween them or else between one of them and a proper initial segment of the other. This construction will employ a family of three classes of symbols at each step in the recursion.

For the foundation family, F_0, we initialize $X_0 = X$, $Y_0 = Y$, and $P_0 = ()$, the empty class. Then we apply the successor algorithm.

Set x_0 to be the least element of X_0. Set $X_1 = difference < X_0, x_0 >$, which is X_0 with its least element removed.
Set y_0 to be the least element of Y_0. Set $Y_1 = difference < Y_0, y_0 >$.
Set $P_1 = (< x_1, y_1 >)$, which is the union of P_0 and $< x_1, y_1 >$.
The successor family is $F_1 = X_1, Y_1, P_1$.

The general recursion at F_k is
Set x_k to be the least element of X_k.
Set $X_{k+1} = difference < X_k, x_k >$, which is X_k with its least element removed.
Set y_k to be the least element of Y_k.
Set $Y_{k+1} = difference < Y_k, y_k >$.
Set $P_{k+1} = union(P_k, < x_k, y_k >)$, which is the union of P_k
and $< x_k, y_k >$.
The successor family is $F_{k+1} = X_{k+1}, Y_{k+1}, P_{k+1}$.

This recursion is assumed to be carried out to completion, which is the point at which no new pairs can be created.

There are only three possibilities at completion:

1) Both residual classes derived from X and Y are empty. Therefore all of their elements now occur in the pairs in P and we have created an order isomorphism between X and Y.

2) Residual X is empty but residual Y is not empty. Therefore we have created an order isomorphism between X and a proper initial segment of Y.

3) Residual Y is empty but residual X is not empty. Therfore we have created an order isomorphism between Y and a proper initial segment of X.

We have assumed that X and Y are well ordered classes of symbols. Therefore, the proof need not employ the axiom of choice.

The weak point of this proof is the assumption that the recursion can be completed, even when it is not a finite recursion. This is, however, not nearly as daring as a proof in ZFC, in which it is axiomatically assumed that every set can be well ordered (one form of the axiom of choice) and, however great the ordinalities are, the recursion can be completed. This **axiom of choice** is believed to be independent of the ZF axioms but it is counterintuitive in view of the fact that we don't know how to well order the mostly undefinable elements of any transdenumerable sets. By contrast, any well defined class of symbols of a finite alphabet can be canonically well ordered. If the alphabet is finite, all of its symbols can be finitely generated in canonical order. If an algorithmic filter is available for some subclass, the same can be done for that subclass.

Nevertheless, we know that some classes of symbols can be well ordered so as to have infinitely many ordinal limit points. An example of this is the class of symbols in the form $< x, y >$, where x and y are any natural numbers and we assume that the order of this class is lexicographic. This ordered class cannot be created by a finite recursion. It's extension can be finitely created but in a different order.

This suggests that the definition of well definedness should include a class of ordered symbols if all of its elements can be finitely generated and the order can be textually defined, even if that is not the order of creation.

31. Author's Notes

7/05/2015: This tutorial is being published online without having been reviewed. It is still subject to editorial changes and it may contain errors or important omissions. (I apologize for this but I am aware that Georg Cantor died at the age of 72 after a prolonged mental illness. Being 90, I am in a bit of a hurry.) Comments may be sent to the author at email: symbolicsets@yahoo.com.

Suggestions that lead to material changes will be used with attribution.

I am grateful to Alexander Karnaugh (of Minsk) for his help in formatting this document with the LATEX language and to John Kastner (a Californian) for his generous aid in getting it online. I am deeply indebted to my wife, Linnie, for her unwavering patience and encouragement during the creation of this work.

1/17/2017: I have now edited this paper to make it a little more accessible.