

## THE CLASS OF N-BITS: A STRUCTURE FOR THE CREATION OF BINARY TRANSMEDIA

We are beginning to understand the methods and structures by which poetry, visual art, music and other types of media are able to continuously transmute themselves within and across forms.

The mathematician Georg Cantor was able to prove that there are not only infinite sets or infinite collections of things, but that there are different types of infinities. Some infinities are larger than others. Once Cantor proved that there were different types of infinities, the next step was to create a single set containing all possible subsets of every type of infinity. Such a set would provide a syntactic structure or container capable of holding every possible type of semantic content. Cantor searched for this set and eventually came to believe that it did not exist.

The search did not begin with Cantor. The search for a perfect language, a language able to describe all things, has over thousands of years led to a world filled with languages, an almost infinite number of languages. Every person, every category of interest, every point of focus, every thing and every thing associated with every thing has become a language in and of itself. Language has become synonymous with identity. But identity cannot create. It can only rearrange its parts until it begins to take on the character of a contortionist whose entertainment value and consequent success is measured by the ways in which the various body parts can be twisted, turned, dislocated, drawn close together or separated far apart. No matter how good the performance, eventually we get bored. And in our boredom, language becomes the enemy of art.

That is why artists, and poets in particular, are increasingly intrigued with the possibility of creating art that is continuous and dynamic, that creates new, concrete instances of itself without the involvement of its author, that transmutes from one artistic form to another, that invites the active intervention of other art and artists, that moves freely throughout space and time. In short, they want to make art that looks and sounds and acts like the world around them. In this way, what seemed a mechanistic, combinatorial divergence becomes instead a creative convergence, a language of languages, dependent on others for its identity.

### *The Omega Class*

For any ordered set (i.e. class)  $S$  of symbols there exists a class  $\Omega$  (called Omega) of  $N$  (i.e. variable) length symbol strings similar to the set of natural numbers  $\{1, 2, 3, \dots\}$  where  $\Omega$  is further defined as the class of all ordered combinations of  $N$  length symbol strings where:

$$N = \{1, 2, 3, \dots\}$$

For example, where

$S = \{0, 1, 2\}$

$\Omega = \{0, 1, 2, 00, 01, 02, 10, 11, 12, 20, 21, 22, 000, \dots\}$

Or where

$S = \{a, b, c\}$

$\Omega = \{a, b, c, aa, ab, ac, ba, bb, bc, ca, cb, cc, aaa, \dots\}$

Furthermore, the class  $\Omega$  is equivalent for all  $S$ .

The Class  $\Omega$  of  $N$ -bits

The class  $\Omega$  in binary form would be represented as:

$\{0, 1, 00, 01, 10, 11, 000, \dots\}$

This is the class  $\Omega$  of  $N$ -bits (where bit means *binary digit*).

Furthermore, the binary representation of members of the class  $\Omega$  of  $N$ -bits does not require that any member be interpreted as a number value. In other words,  $\Omega$  is a syntactic structure as opposed to a semantic system. There is nothing to prevent a member of the class  $\Omega$  from being identified with a member of any type (of class). As a result the class  $\Omega$  is capable of defining any number of like and unlike types as subclasses.

When interpreted as a class whose members are the natural numbers, the class  $\Omega$  is shown to contain a transfinite (i.e. infinite) number of each member of the class of natural numbers. For example, the binary number 1 is represented by the subclass  $\{1, 01, 001, \dots\}$ , the binary number 2 by the subclass  $\{10, 010, 0010, \dots\}$ , etc.

The class of bytes is defined as a subclass of the class  $\Omega$ :

$\{00000000, 00000001, 00000010, \dots, 11111111\}$

*Syntax and Semantics*

By definition, the class  $\Omega$  of  $N$ -bits is the parent class of all classes whose members may be represented as binary symbol strings. A member of  $\Omega$  does not represent a specific object or number value. The member is nothing more than a binary symbol string – a syntactic, abstract symbol over which the semantic "template" or "idea" of the object is placed. This is another way of saying that the information is not contained in the data. The information results from the model (or template or idea) used to interpret the data.

And (this is very important), that template or idea is itself determined by the relationship of the original symbol string to other symbol strings, also members of the class  $\Omega$ .

By specifying the length in bits and the binary values of members of the class  $\Omega$  of  $N$ -bits as a subclass, it is possible to define classes of any type commonly used in computing. Subclasses of the class  $\Omega$  include the natural and real number systems, microprocessor instruction sets, virtual machine instruction sets and objects of any type (including state models, state machines, texts, images, sounds, etc.). Simply stated, the class  $\Omega$  of  $N$ -bits provides a means whereby all classes, attributes and relations representable in binary or digital form may be defined as members of  $\Omega$  or its subclasses.

### *Relational Attribution*

As with any class, the use of ordered pairs (i.e. Cartesian products) provides an opportunity to define relations within class definitions. A relation is a class whose members are ordered pairs that share an implicit or explicit relationship that is equivalent for all member pairs. An attribute is an abstraction of a characteristic shared by all members of a class. Relational attribution is a way of creating attributes from relations. The relation may be simple or complex and may take the form of one or more logical operations and/or mathematical functions.

Several examples can demonstrate the varying complexity of these relations. One example is the class Parent/Child whose members are Mother/Son, Mother/Daughter, Father/Son and Father/Daughter. Another example is the "Fibonacci sequence" 1, 1, 2, 3, 5, 8, 13, 21, etc. where each succeeding number after the first two is the result of adding the two numbers immediately preceding it. A third example is the series of "random" digits 5, 8, 9, 7, 9, 3, 2 that represent the 11-17<sup>th</sup> digits in the calculation of Pi.

The concept of relation is used to provide class membership to all functions (i.e. the notion of  $\{xy: Fxy\}$ ) that designate the relation of anything  $x$  to anything  $y$  such that there exists a function that describes a relation between the values (i.e.  $Fxy$ ). Whether the relations are implicitly or explicitly defined, the use of dyadic (i.e. two), triadic (i.e. three) and higher relations allows us to define both natural and real number systems as subclasses of  $\Omega$ . For example, a data type representing integer values of arbitrary precision may be defined as the following triadic relation:

S (1 bit) followed by L ( $N$  bits) followed by V(  $L$  bits)

where

S is the sign bit,

L is a binary value representing the length in bits of the binary representation of the number,

$N$  is an implementation specific number of bits greater than zero defining the length in bits of  $L$  and

$V$  represents the binary value of an integer of length  $L$  beginning with the least significant bit.

An example of this  $N$ -bit data type is:

11100010110100110101110101101001010000101010100101101111

where

$S = 1$  (1 1100010110100110101110101101001010000101010100101101111),

$L = 110001$  (1 110001 0110100110101110101101001010000101010100101101111, i.e. 49)

$N = 6$  (i.e. the length in bits of  $L$ ) and

$V = 0110100110101110101101001010000101010100101101111$  (1110001 0110100110101110101101001010000101010100101101111)

A data type representing floating point values of arbitrary precision may be defined as follows:

$S$  (1 bit) followed by  $L$  ( $N$  bits) followed by  $R$  ( $N$  bits) followed by  $V$  ( $L$  bits)

where

$S$  is the sign bit,

$L$  is a binary value representing the length in bits of the binary representation of the number,

$N$  is an implementation specific number of bits defining the precision,

$R$  identifies the radix point and

$V$  represents the binary value beginning with the least significant bit.

The length in bits of  $L$  for integer values and of  $L$  and  $R$  (for floating-point values) are implementation specific and in practice limited only by available real memory (i.e. the RAM or address space of a computer). For example, if  $L$  and/or  $R$  are defined as being 24 bits in length, integer and floating-point values with up to 16,787,456 significant binary digits are possible.

If we describe integer and floating-point values using these class definitions, the use of binary adders to perform digital addition, subtraction, multiplication and division and subsequently any operation in higher mathematics is greatly simplified. Although there are certain operations (i.e. division) and certain numbers (i.e. Pi) that result in an infinite series or sequence of binary digits, these data types greatly reduce the possibility of rounding and overflow errors.

### *Cardinality*

In 1874 Georg Cantor proved that the cardinality of the set of natural numbers (i.e. positive integers) was not equivalent to the cardinality of the set of real numbers (i.e. points on a line segment). The cardinality of the natural numbers he called aleph-null. The cardinality of points on a line segment he referred to with a lower case German *c* for "continuum." The cardinality of the natural numbers was countable. The cardinality of the set of real numbers was not. This proof established the foundation for what is now referred to as Cantorian set theory.

Although the cardinality of the class  $\Omega$  is identical to that of the natural numbers (i.e. Cantor's aleph-null), it can be demonstrated that the creation of floating point numbers of arbitrary precision is possible using *n*-adic relations. The class  $\Omega$  is countable, but relations on  $\Omega$  may be used to describe classes whose cardinality is uncountable (e.g. the real numbers) and yet whose members are expressible as members of the class  $\Omega$  of *N*-bits.

As an example, we may modify our current definition of floating point values to include transfinite precision. (Transfinite is a term Cantor used to describe to describe cardinalities that lay somewhere between the finite and the infinite):

$L(N \text{ bits})$  followed by  $R(N \text{ bits})$  followed by  $V(L \text{ bits})$

where

*L* is a binary value representing the length in bits of the binary representation of the number,

*N* is a binary value of transfinite length that defines the precision of the floating point value,

*R* identifies the radix point and

*V* represents the binary value.

If the concept of a transfinite value (i.e. *N*) seems to violate the intent of the above example, one can as easily treat *N* as a countable series (e.g. 1,10,11,100, . . . ) of binary values that extends the precision of the floating point values with the same result.

The value of  $N$  (i.e. the precision) has cardinality aleph-null (i.e. the cardinality of the natural numbers) and yet the method provides a means of expressing any real number as a member of the class  $\Omega$ . It is a simple matter to show that for any real number value derived using this method there is a corresponding natural number expressible as an  $N$ -bit value. For example, by using one or another of the above methods, the binary symbol string "100" can be interpreted as either the natural number "4" or the floating point number ".0" .

What this means is that  $\Omega$  contains an infinite number of each member of the set of natural numbers and an infinite number of each member of the set of real numbers. This raises questions regarding the cardinality of  $\Omega$  since for any real number defined using this method there is a corresponding natural number.

Another way to compare the cardinality of the natural numbers with the cardinality of the real numbers is to try to find a real number that is not expressible as a member of the class  $\Omega$  of  $N$ -bits. One method that has been used to show that the set of natural numbers and the set of real numbers are not the same is to look at the space between any two real numbers as an infinite set. For example, if we look midway between the natural numbers 1 and 2 we can find the real number 1.5. Then by looking between 1.5 and 2, we can find the real number 1.75. No matter how many times we subdivide the space between two numbers we will always have an infinite set of numbers remaining. Although the real numbers do not occur in sequence in the class  $\Omega$  of  $N$ -bits they do occur in the class. In fact, each and every natural number and each and every real number can be shown to occur an infinite number of times within the class  $\Omega$  of  $N$ -bits.

If we think of the class  $\Omega$  as representing simple integer values we notice that each integer value is represented by an infinite sequence (i.e. {0, 00, 000, . . . }, {1, 01, 001, . . . }, etc.) of  $N$ -bit values. In fact, each point on a line segment (represented as a floating-point value) is also expressible as a transfinite series of  $N$ -bit values. This again raises questions regarding the cardinality of  $\Omega$ . If the class of natural numbers with cardinality aleph-null and the class of real numbers with cardinality  $c$  (i.e. the cardinality of the real numbers) are both expressible as subclasses of the class  $\Omega$  of  $N$ -bits, then what is the cardinality of  $\Omega$ ? It appears that  $\Omega$  is a class with the same cardinality as that of the natural numbers capable of representing all of the values of the real number system having cardinality  $c$ . In other words, the cardinality of  $\Omega$  is not evident in its syntactic form, but is determined by its semantic use.

Since the cardinality of  $\Omega$  is a function of the model used to interpret its members and since the same class used to define both natural and real numbers may also be used to represent the rules for the formation of all possible subsets of itself, it follows that all cardinalities are definable within  $\Omega$ .

### *Predicate Logic and Object-Oriented*

In 1899 Giuseppe Peano became the first mathematician to use symbolic logic to define the axioms of arithmetic. The same method was used by Zermelo-Fraenkel-Skolem to

create the axiom system for set theory. Symbolic logic provided a means whereby any mathematical text could be expressed as a series of well-formed statements in a formal language. The formal language that defines the variables, axioms and operations of arithmetic (i.e. addition, subtraction, multiplication, division, etc.) using the terms of symbolic logic is referred to as Peano Arithmetic. Peano Arithmetic, in turn, can be extended to provide a formal definition for higher mathematics.

In 1930 Kurt Gödel demonstrated that it was possible using a technique called Gödel numbering to translate statements in any formal language into equivalent expressions (number values) that obeyed the rules of Peano Arithmetic.

Very simply, Gödel numbering creates a one-to-one correspondence between any set of symbols and members of the set of natural numbers  $\{1,2,3, \dots\}$ . As an example, a scheme for Gödel numbering the symbols of axiomatic set theory is:

~	1	[	11	<i>p</i>	21
∨	2	]	12	<i>q</i>	22
&	3	<i>S</i>	13	<i>x</i>	23
→	4	+	14	<i>y</i>	24
↔	5	×	15	<i>z</i>	25
∃	6	=	16	...	...
∀	7	<	17		
(	8	>	18		
)	9	0	19		

Any string of symbols in any language so defined can be converted into a corresponding Gödel number by substituting each symbol in the string into its corresponding number value and separating the symbol with the number 0.

For example, the statement:

For every  $x$  there is a  $y$  such that  $x < y$

stated in axiomatic set notation as

$(\forall x)(\exists y)(x < y)$

can be converted into the Gödel number

8 0 7 0 23 0 9 0 8 0 6 0 24 0 9 0 8 0 23 0 17 0 24 0 9

or

807023090806024090802301702409.

Gödel numbering also allows for the inclusion of data within the body of a mathematical text. Kurt Gödel's achievement created a consistent and rigorous chain of logic leading from the axioms of simple arithmetic to well-formed statements (statements provable from a set of axioms) in any formal language. Since it contains all possible combinations of binary symbol strings, the class  $\Omega$  of  $N$ -bits may be employed as a means of assigning (as opposed to constructing) Gödel numbers to any and all possible classes, attributes, relations and statements (including propositional and predicate texts). In so doing, the class  $\Omega$  becomes the parent class of those classes, attributes, relations and statements.

Axiomatic set theory as it has evolved through the work of Cantor, Zermelo, Russell, Gödel, von Neumann and Quine has resulted in the ability to define attributes, open sentences, propositional functions (in the form of attributes or predicates) and relations of like and unlike type as classes. Classes, in turn, can be made to serve as the foundation for all logical and mathematical texts. This means that all formal grammars can be represented as members of  $\Omega$ .

It is also possible, using the class  $\Omega$  of  $N$ -bits, to create subclasses of data structures and/or relations that may be applied to an infinite number of heterogeneous class types, either of data or of other relations. This is another way of saying that any object, including its data structures, relations and behavior, can be represented as a member of  $\Omega$ . This means that it is possible to use inheritance from class types of data structures and/or class types of relations (i.e. propositional and/or predicate texts) as a means of creating new, previously undefined class types of data and/or relations.

Furthermore, since for any ordered set (i.e. class)  $S$  of symbols there exists a class  $\Omega$  and since  $\Omega$  is equivalent for all  $S$ , any text derived from any  $S$  may be represented as a member of  $\Omega$ .

### *Entropy and Compression*

The term entropy as it is used in information theory is a measure of how much information is contained or encoded in a message. A message in turn is defined as a string of symbols. The higher the entropy of a message, the greater is its information content. The lower the entropy, the smaller is its information content. Data compression is to information theory what set theory is to higher mathematics and as such becomes the means by which we understand the fundamental nature of information.

Information theory teaches us that there is no absolute measure of the information content of a symbol string. The assumption that the content or meaning of any data is somehow contained within the digital representation of the data is incorrect. The information content of any data is a function of the model used to interpret that data.

The class  $\Omega$ , for example, is essentially a collection of uninterpreted symbols. A string of symbols (i.e. bits) has little or no meaning in and of itself. Its meaning is extrinsic. By extrinsic is meant that its meaning is dependent on relations established between it and other binary symbol strings that are also members of  $\Omega$ . In other words, for every

member  $x$  of  $\Omega$ , there exists an infinite number of subclasses of  $\Omega$  that define relations on  $x$  with other members of  $\Omega$ . These other members (i.e. symbol strings) may take the form of numbers, functions, objects, models, well-formed statements in a formal grammar, or entire formal systems. It is the defined relations of any member of  $\Omega$  with other members of  $\Omega$  that determine its semantic interpretation. Given an appropriate model, any member of  $\Omega$  may be substituted for any other member of  $\Omega$ . When the substitution results in a shorter symbol string than the original, lossless compression is possible. Lossless compression means that any data, once compressed, can be returned to its original state without the loss of a single bit.

The class  $\Omega$  of  $N$ -bits provides a means whereby the entropy (i.e. information content) of a given data set may be increased or decreased at will regardless of the original content of the message.

One way this can be accomplished is by varying the length in bits of the input symbols to the model (i.e. program) used to compress the data. This, in effect, changes the model. So, in practice, the input symbols can be varied until desired entropy is obtained. Changing the length in bits of the input symbols to find the desired entropy allows data that was previously considered to be uncompressible to be compressed. Data compressed at some bit length will reach an entropy limit and converge at the calculated probabilities for that data set. To change the entropy limit and the calculated probabilities, one has only to change the model.

The entropy of a given symbol in a symbol string is defined as the negative logarithm of the probability of its occurrence in the string. The entropy of a symbol string (i.e. message) is defined as the sum of the entropy for all the symbols in the string. The formula for determining the entropy of a given symbol in a binary message is:

Entropy (i.e. number of bits) =  $-\text{Log base 2 (number of like symbols/total symbols in message)}$

As an example, the binary value 01000010 is a standard representation of the letter "B" using the American Standard Code for Information Interchange (i.e. ASCII). The binary value 01000010 means "B" only because it was decided by a committee at some in time to represent the English alphabet using eight bits and 01000010 was designated the letter "B".

By changing the length in bits of the input the entropy for the letter "B" can be made to vary:

Symbol	Probability	Entropy
01000010	1/1	<u>0</u>
Total =	0	
0100	1/2	1

0010	1/2	<u>1</u>
Total =	2	

01	1/4	2
00	2/4	1
10	1/4	<u>2</u>
Total =	5	

It is believed by many that random binary data is not compressible. All binary symbol strings can be shown to have an entropy lower than their length in bits. The key is to use an appropriate model. For example, by interpreting the following quantum mechanically derived random binary string of sixty-four bits as a series of two bit words, an entropy value lower than the original sixty-four bits is made possible.

Given the eight bytes (64 bits)

```
00100101 00000110 00001100 10000111
01011100 01110011 00011100 01011100
```

the two bit words (i.e. symbols) are

```
00 10 01 01 00 00 01 10 00 00 11 00 10 00 01 11
01 01 11 00 01 11 00 11 00 01 11 00 01 01 11 00
```

“00”	= 16.98045
“01”	= 16.78072
“10”	= 10.24511
“11”	= 15.34852
Total	= 59.3548 (bits)

Example:  $-\text{Log base } 2(12/32) = 16.98045$   
(given 32 total symbols and 12 “00” symbols)

The base size of the binary symbol set may be modified to optimize the compressibility of any source of binary input regardless of whether that source of input is also the output of a previous compression process. There is no theoretical limit to the number of times a symbol string may be repeatedly compressed. This is because the entropy of a symbol string results as much from the model (i.e. semantic context) used to interpret the string or message as it does from the string (i.e. syntactic structure) itself.

In fact, it is possible to construct a very simple grammar capable of substituting any symbol string for any other symbol string. For example a language  $L$  could be defined as:

$$L: \quad a \rightarrow b$$

$$\quad \quad b \rightarrow a$$

where

$a$  is any symbol string,  
 $b$  is any symbol string other than  $a$ ,  
 $a \rightarrow b$  is interpreted as "for  $a$  substitute  $b$ " and  
 $b \rightarrow a$  is interpreted as "for  $b$  substitute  $a$ ."

By placing a program containing these rules at either end of a communications line, we have a method of "compressing" any symbol string into any other symbol string. For example, we can let  $a$  equal the contents of Herman Melville's *Moby Dick* and  $b$  equal "0". By translating an arbitrarily large string of data into an arbitrarily small string of data we enable almost infinite compression.

### *Transformational Grammars*

Noam Chomsky has proposed that the human mind's innate capacity for language may be described in terms of a "universal grammar." This universal grammar is generative in nature and consists of a finite set of canonical forms (i.e. concepts) and a finite set of rules that act upon these forms. This combination of forms and rules describes a system capable of representing the deep structure of all human knowledge. Chomsky suggests, following Kant, that this universal language has an a priori aspect based upon species specific genetic codes. He does not, however, prescribe the relative roles that innate capacity (i.e. genetics), sensory input and socially defined behaviors play in the construction of this language.

Although the forms that constitute the "objects" of language have both physiological and psychological characteristics, they also have schematic representations. These schematic representations (i.e. "artifacts" that take the form of speech or writing) define the structural components of language. The transformational (i.e. behavioral) components of language consist of rules that effect changes in the occurrence and order of the artifacts. It is the combination of these artifacts and the rules that act upon them that constitute the language of a given knowledge domain.

An infinite number of grammars may be defined to describe those data transformations that may be used to manipulate the order and occurrence of symbols within any symbol string derived from any ordered set (i.e. class)  $S$  of symbols. The following notation describes one such grammar for the class  $\Omega$  of  $N$ -bits:

$\Sigma:$      $Z$   
 $F:$      $Z_1 \rightarrow a + b$   
            $Z_1 \rightarrow a + Z_2 + b$

where

$\Sigma$  represents a finite set of binary symbol strings and

$F$  represents a finite set of symbol substitution rules of the form  $Z_1 \rightarrow a + b$

interpreted as "for any binary symbol string  $Z_1$  that is a member of  $Z$ , rewrite binary symbol string  $Z_1$  as binary symbol strings  $a + b$ " where  $a$  and  $b$  represent adjacent substrings of  $Z_1$  inclusive. The operator "+" is defined as concatenation.  $Z_2$  is defined as any element of  $Z$ .

Insert

$\Sigma:$   $Z$   
 $F:$   $Z_1 \rightarrow a + b$   
 $Z_1 \rightarrow a + Z_2 + b$

Delete

$\Sigma:$   $Z$   
 $F:$   $Z_1 \rightarrow a + b$   
 $Z_1 \rightarrow a$

Substitute

$\Sigma:$   $Z$   
 $F:$   $Z_1 \rightarrow a + b$   
 $Z_1 \rightarrow a$   
 $Z_1 \rightarrow a + Z_2$

Append

$\Sigma:$   $Z$   
 $F:$   $Z_1 \rightarrow Z_1 + Z_2$

These primitive methods may be used in combination to describe more complex machine translation rules:

Swap

$\Sigma:$   $Z$   
 $F:$   $Z_1 \rightarrow a + b$   
 $Z_1 \rightarrow b + a$

Rotate

$\Sigma:$   $Z$   
 $F:$   $Z_1 \rightarrow a + b + c$   
 $Z_1 \rightarrow c + a + b$

Interleave

$\Sigma:$   $Z$   
 $F:$   $Z_1 \rightarrow a + b$   
 $Z_2 \rightarrow d + e$   
 $Z_3 \rightarrow a + d + b + e$

These same primitive methods may also be used to define translation rules for natural language grammars.

Just as it is possible, using the class  $\Omega$  of  $N$ -bits, to derive all possible classes of data structures and relations, it is also possible to define any transformational grammar in

terms of  $N$ -bit structures and operations. The class  $\Omega$  of  $N$ -bits may be used to describe both the artifacts and rules for any given knowledge domain.

*The Class  $\Omega$  as Foundation Class*

In summary, the class  $\Omega$  contains transfinite (i.e. infinite) numbers of classes, attributes, relations, variables and data types as proper subsets of itself. Since for any member  $x$  of  $\Omega$ , there exists an infinite number of subclasses that define relations on other members of  $\Omega$ , all possible texts in any formally defined grammar may be expressed as members of the class  $\Omega$  or its subclasses. As a result, the class  $\Omega$  of  $N$ -bits contains an environment rich enough to represent any statement in any formal language as a combination of binary values and relations.

Since the class  $\Omega$  of  $N$ -bits is logically equivalent to any instance of the class  $\Omega$  derived from any ordered set  $S$  of symbols, the class  $\Omega$  of  $N$ -bits also contains the symbols and rules necessary to transform binary expressions into equivalent expressions in any class  $\Omega$  for any ordered set of symbols  $S$ . This means that the class  $\Omega$  of  $N$ -bits identifies a foundation class for the derivation of all possible grammars and their manifestation within the context of a global, digital computing network.

*This article was adapted from the artist's book Red Moon by Michael Harold.*