Introduction to Dynamic Unary Encoding

Ernst D. Berg
ernst@eberg.us
Turlock, CA.

December 19, 2014

Abstract

Dynamic Unary Encoding takes Unary Encoding to the next level. Every \( n \)-bit binary string is an encoding of dynamic unary and every \( n \)-bit binary string is encodable by dynamic unary. By utilizing both forms of unary code and a single bit of parity information dynamic unary encoding partitions \( 2^n \) non-negative integers into \( n \) sets of disjoint cycles of \( n \)-bit elements. These cycles have been employed as virtual data sets, binary transforms and as a mathematical object. Characterization of both the cycles and of the cycle spectrum is given. Examples of encoding and decoding algorithms are given. Examples of other constructs utilizing the principles of dynamic unary encoding are presented. The cycle as a mathematical object is demonstrated.

1 Introduction

Dynamic Unary Encoding (DUE), in the form of a binary data encoder, was discovered by this author in January of 2010 during an effort to satisfy a data compression challenge through experimental binary encoding schemes involving iterative functions. Although the effort to find a solution to that challenge continues; DUE is a discovery of importance. Effort is made to present DUE in an introductory format leading to suggestions of applications in higher mathematics and computer science. It has not escaped observation that Dynamic Unary has applications beyond data encoding.

1.1 Overview

The history and development of Dynamic Unary Encoding is a reflection of my search for innovative binary encoding resulting in four categories of algorithms developed through experimentation and observation.

The first category of encoder developed is introduced as an iterative function exporting the parity of the first bit, in the binary string being encoded, as the transform. An example of this category of algorithm is presented in 5.1.

The second category includes binary string constructions and deconstructions exploring the relationship of parity information and unary code to binary string structure. An example is presented in 5.2.

The third category communicates the parity structure of the source string via the terminus parity of the unary code. Through iteration of encoding or decoding it was observed that for all \( n \)-bit binary strings DUE generated a cycle of \( n \)-bit elements.

For the fourth category the cycles of DUE as a complete orbit are successfully utilized as a mathematical object; a type of dynamic integer or Quantum Numbers perhaps? A non-DUE cycle is generated as the example of this category.
1.2 Unary code

Unary Code[1] is an entropy code that represents a non-negative integer. Unary code has the form of a body of zero or more bits of one parity and a single bit terminus of the opposite parity.

- Example: Integer value four can be represented in unary code as both 0111 and 100[1].

A finite length binary string may contain more than one unary coded integer. Each integer is delimited by the terminus parity.

- Example: In the binary string “011101110110100” unary codes with a terminus of the 0 parity represent integers \{4, 5, 3, 2, 1, 1\}.

1.3 Parity structure of binary strings and the Parity Reference

Binary strings have the quality of being dividable into segments of same parity bits in an alternating parity pattern forming the parity structure of the binary string.

- Example: For “011101110110100” same parity bit segments are \{0, 111, 0, 1111, 0, 11, 0, 1, 00\}.

This parity structure is communicated with a single bit by knowing the parity of a single bit at a specific bit position. The term for this is the Parity Reference. The term Parity Reference incorporates both the physical location of that bit and the information of that bit’s parity state. The Parity Reference may be internal to the encoding or can be external. Bit position notation follows conventional representation of \(b_0\) (first bit) through \(b_{n-1}\) for some finite length binary string. A finite length binary string is also referenced as an \(n\)-bit.

2 Encoding and Decoding

The Cycle and Mathematical Object aspects of Dynamic Unary Encoding are presented first followed by two encoding constructs utilizing external Parity Reference constructs. Dynamic Unary Encoding utilizes Unary Code and Parity Reference information for all constructs.

2.1 Encoding

First, encoding parses a finite length binary string into same parity segments. The terminus parity of the unary code is determined by the parity of the bit at the Parity Reference bit position. Once the binary string parity structure is exported as terminus parity a simple bijection of segment length and unary code exists. The lengths of those segments are then represented in a new string by unary codes. An example encode algorithm is presented in figure 1.

- Let \(\sigma\) represent encoding.
- Let our source string be “0100100001100101011011000110110001101111” “Hello” in ASCII code.
- Let the Parity Reference = \(b_0\) making the unary code terminus = 1.

Same parity segments 0 1 00 1 0000 11 00 1 0 1 0 1 1 0 11 00 0110 0 11 0 000 11 0 11 000 11 0 1111

Unary Encoded 1 1 10 1 1000 10 10 1 1 1 1 10 1 10 100 10 1 10 100 10 1 1000

\[\sigma(0100100001100101011011000110110001101111) = (111011000101011111011010010111001011000)\]

1Binary data are presented in the Western Positional Notation of right to left.
2.2 Decoding

First, decoding determines the terminus parity of the unary code used to encode from the $n$th ($b_{n-1}$) bit of the string. Then the string is parsed into unary codes by terminus boundaries. Same parity segments, of the length the unary code communicates, are then written in correct parity order using the Parity Reference information. An example decode algorithm is presented in figure 2.

- Let $-\sigma$ represent decoding.
- Let our source string be “11101100010111110110100101101001011000”
- Let our unary code terminus be 1 and our Parity Reference = $b_0$. Terminus information is read at $b_{n-1}$.

Parse the string into unary codes $1 1 0 1 1 0 0 0 1 0 1 0 1 1 1 1 1 0 1 0 1 1 0 0 1 0 1 0 1 0 1 1 0 0 0 1 1 0 1 1 0 1 1 1 1$

$\rightarrow 1 0 0 1 0 0 0 0 1 1 0 0 1 0 1 0 1 1 0 1 1 0 0 1 1 0 1 1 1 1$

$-\sigma(11101100010111110110100101101001011000) = (010010000110010110110001101100011111)_{10}$

3 Cycles and Cycle Spectrum

Every $n$-bit binary string is an encoding of dynamic unary and every $n$-bit binary string is encodable by dynamic unary. Through experimentation and observation iteration of encoding or decoding resulted in a cycle of $k$ elements of length $n$ bits. By utilizing both forms of unary code and the parity structure information of the source string dynamic unary partitions $2^n$ non-negative integers into disjoint cycles of $n$-bit elements creating a dynamic data type. The spectrum of these cycles is an infinite progression of sets of cycles that have in common relationships to powers of two. The quality of the cycles being disjoint was first determined by observations of the cycles of the shorter length strings which are easy to generate by hand. The disjoint quality of the cycles was also observed with a C-Language program generating the cycles of thirty two bit elements, as a data set, of all $2^{32}$ non-negative integers. Encoding is further defined as generating the next element in cycle and decoding is further defined as generating the previous element in cycle.

$$S_0 = \{0000\} \quad n = k = 8 \quad PRef = b_0 \quad S_{i+1} = \sigma(S_i) \quad or \quad -\sigma(S_i) \quad for \ 0 \geq i < k \quad S_k = S_0$$

$\sigma(0000) = (01111100010111110000110100000) = (71251583100)$

$-\sigma(0000) = (101000111000101111011000111100000) = (10381551270)$

$S_0 \rightarrow S_1 \rightarrow S_2 \rightarrow S_3 \rightarrow S_4 \rightarrow S_5 \rightarrow S_6 \rightarrow S_7 = S_0$

Example cycle in both the encode direction and the decode direction of a 4-bit string.

The number of cycles for a n-bit binary string and a specific Parity Reference bit position is determined by $x$ equals the number of elements in a cycle and $2^n$ the number of non-negative integers such that $cycles = 2^n / x$. That there are $n$ choices for a Parity Reference means we also have $n$ sets of cycles for each $n$-bit string.

- From observations of the cycle data, cycle spectrum can be separated into three categories.
- String length $(2^n)^{b_0}$ has $(2^{1+n})$ cycles of $(2^n)$ elements for $n = 0$.
- Binary string lengths that are a power of two, greater than one, and Parity Reference of $b_0$ have been observed as having an element count of $(2^{1+n})$ and cycle count of $(2^{2^n}/2^{1+n})$ for $n \in \{1, 2, 3, \ldots\}$.
- All other lengths and Parity Reference combinations may be defined as $(2^{1 + \lfloor \log_2 n \rfloor})$ elements and $(2^{1+n}/2^{1+\lfloor \log_2 n \rfloor})$ cycles for $n \in \{1, 2, 3, \ldots\}$.

\[A complete listing of the cycles for string lengths one through eight is presented in table one.\]

\[In researching the Mathematics of Cycles it was observed that a cycling of a single element is not generally considered a Cycle.\]

\[However, what applies to a string of one bit, in DUE, applies to all strings and is therefore included as such.\]

\[\{2^{1+\lfloor \log_2 n \rfloor}\}_{n \in \{1, 2, 3, \ldots\}} = \{2, 4, 8, 8, 8, 8, 16, 16, 16, 16, 16, 16, 32, 32, 32, 32, \ldots\}\]
4 The Cycle as a mathematical object

4.1 Introduction

By utilizing the complete orbit, application as a mathematical object is realized. As a demonstration a non-DUE cycle is presented. The concepts of “Cycle-Of” ($\sigma_{Of}$) and “Cycle-On” ($\sigma_{On}$) are now introduced to differentiate types of cycles. Cycle-Of is defined as the dynamic unary cycle of an element. Cycle-On is defined as applying Cycle-Of to an element. In the following example a function utilizing the orbits of three integers and the logical operation exclusive-or ($\oplus$), a ($\sigma_{On}$) cycle of thirty two elements is generated. That for the element length of sixteen bits and a Parity Reference $b_0$ the number of ($\sigma_{Of}$) elements is thirty two and the number of elements in the ($\sigma_{On}$) is also thirty two. This is not always the case. For example change the Parity Reference to any other bit position and the number of elements in the ($\sigma_{Of}$) orbit is sixteen. Curiously the number of elements in the ($\sigma_{On}$) orbit is still thirty two. Thus for Parity Reference $b_0$ only one ($\sigma_{Of}$) orbit occurs and for all other Parity References two ($\sigma_{Of}$) orbits are required to generate a complete ($\sigma_{On}$) orbit. The halting condition of both the Cycle-Of and the Cycle-On is when the element generated equals the starting element. The reason only one ($\sigma_{Of}$) orbit occurs for string lengths of powers of two and a Parity Reference of $b_0$ is seen in the element structure of the cycle in that the elements of those cycles can be divided into two sets where the members of each are parity inverses of the other.

- An example of length four bits $\sigma_{Of}(0000) = (0111 1100 0101 1111 1000 0011 1010 0000 )$

For the following initial conditions this function generates a ($\sigma_{On}$) orbit for the element $S_0 = 2014$.

- Let the length of string be sixteen bits.
- Let the Parity Reference be $b_0$.
- Let $0 \leq j < 32$ and $0 \leq i < 32$ where $S_{32} = S_0$, $X_{32} = X_0$, $Y_{32} = Y_0$ and $Z_{32} = Z_0$
- Let $X_0 = 1, Y_0 = 99, Z_0 = 6408$ and $S_0 = 2014$
- Let $S_{j+1} = S_j \oplus \sigma(X)_i \oplus \sigma(Y)_i \oplus \sigma(Z)_i$

The Cycle-On cycle : $\sigma_{On}(2014) = ( 28158 19761 64921 60058 30232 23332 8057 63754$

27712 19536 951 60323 34882 23123 57674 2015
37376 19760 615 60059 35302 23333 57479 63755
37822 19537 64585 60322 30652 23122 7860 2014 )

If we examine the mapping it is clear that the next element in the ($\sigma_{On}$) cycle is not a result of Dynamic Unary Encoding. $00000111110111110 \mapsto 0110110111111110$. The value 2,014 is mapped to the value 28,158. A quality of this ($\sigma_{On}$) orbit is that by knowing one element in the cycle and it’s index the original element ($S_0$) can be retrieved.

This example utilizes cycles of the same type, direction of spin and element count for simplicity of demonstration however constructs are not limited to same type of cycle, same number of cycles in a function, same spin direction or the cycles being of the same $k$ size.
4.2 Observations

It has been observed that the cycles with element lengths, greater than one, that are a power of two of Parity Reference \( b_0 \) and powers of two plus one of Parity Reference \( b_1 \) have elements, when considered to be unsigned integers, that sum to the same value.

- Let \( \alpha \) represent binary strings length \( 2^n \) and \( \beta \) represent binary strings length \( 2^n + 1 \) for \( 0 < n \leq \infty \)
- \( \sum_{i=1}^{2(2^n)} \sigma(\alpha)_{i}^{b_0} = (2^{2^n} - 1)2^n : \{ \sum \sigma(\alpha) | 6, 60, 2040, 1048560, 137438953440, \ldots \} \)
- \( \sum_{i=1}^{(2^{1 + \lfloor \log_2(2^n + 1) \rfloor})} \sigma(\beta)_{i}^{b_1} = (2^{2^n + 1} - 1)2^n : \{ \sum \sigma(\beta) | 14, 124, 4088, 2097136, 274877906912, \ldots \} \)

Other \( n \)-bit and Parity Reference combinations have been observed as those cycles having more than a single sum suggesting that those cycles may be grouped by a conserved quantity quality.

5 Alternate algorithms

Dynamic Unary has two components, the Unary Code and the Parity Reference information. Each part can be utilized in constructs separately. Two encoding schemes are presented as examples.

5.1 Drop-T

The Drop-T algorithm exports the parity at Parity Reference \( b_0 \) as the encoding transform. The source \( n \)-bit is then encoded in fixed parity terminus unary codes. Each iteration of the encoding then “drops” the \( n \)th bit of the newly unary encoded \( n \)-bit source string before the next iteration. Each iteration reduces the source \( n \)-bit length by one bit and increases output \( n \)-bit length by one bit until all source bits are processed.

Decoding requires that the terminus parity be known. Decoding reads each bit sequentially from the source \( n \)-bit providing the parity information needed to reconstruct the parity structure of the \( n \)-bit binary string of each iteration. Because the construct of encoding is the collection of all the parities of each iteration, Parity Reference is considered external to this \( n \)-bit being constructed. Decoding starts with a single bit unary code and processes it’s parity structure then adds a new terminus bit to the end of that \( n \)-bit increasing the length of the \( n \)-bit string by one bit. This iterative process completes when all the source \( n \)-bit bits are used.

- Let our source be \( \{011\} \) and our Parity Reference be \( b_0 \). Our fixed parity terminus is 1.
  - The encoded string then is all the parities sampled at \( b_0 \)
    \( \{011\} \rightarrow \{110\}, \{10\} \rightarrow \{11\}, \{1\} \rightarrow \{1\} \) so \( \{011\} \rightarrow \{101\} \)
  - It was hoped that Drop-T would be a single bijective cycle for all \( n \)-bit elements representing all \( 2^n \) non-negative integers however it was discovered that some elements maped to themselves.
    - With a terminus parity of 1 and a string of “000” the element maps to itself
      \( \{000\} \rightarrow \{100\}, \{00\} \rightarrow \{10\}, \{0\} \rightarrow \{1\} \) so \( \{000\} \rightarrow \{000\} \)
      Therefore there are more than one cycle for each \( n \)-bit.
- Let our Parity Reference be \( b_0 \). Our fixed parity terminus is 0.
  - For length one the cycles are \( \{(0), (1)\} \)
  - For length two the cycles are \( \{( \ 00 01 10 \ ),( \ 11 \ ) \} \)
  - For length three the cycles are \( \{( \ 000 011 110 \ ),( \ 001 100 010 \ ),( \ 101 \ ),( \ 111 \ ) \} \)
5.2 Binary string construction and deconstruction algorithms

During this phase of development several construction and deconstruction concepts were explored in the hopes of finding some “data encoding magic.” Imagination was key during this phase. The basic constructs of this category process unary code separately from Parity Reference information. Several variations of this type of encoding were explored resulting in unique encoding schemes. For this example, any string length \( n \), two strings of length \( n + 1 \) can be generated. The Parity Reference is an external and fixed parity.

- Given an arbitrary string “011” then adding terminus’ 0+011 and 1+011 then decoding for a fixed Parity Reference of 0 we get \( \{1000\} \) and \( \{0010\} \) because it is assumed the \( nth \) bit is the terminus parity. Therefore it is possible to generate an eight bit string from three bits of information.
- So for this decode \( \{011\} \mapsto \{10000010\} \)

Related work

The cycle structure was first observed by Gustavus J. Simmons\(^5\) and is presented in the paper Parity Encoding of Binary Sequences. Simmons applies an exclusive-or summation algorithm; a “mod 2 sum of the summands”, on binary strings which have in common a finite length and a parity of 1 in the most significant bit position. The cycles generated prove to be a subset of those generated by DUE for the conditions of the decode direction with the Parity Reference bit position of \( b_{n−1} \) when that bit is parity 1.\(^5\)

It is interesting that two different algorithms generate the cycle structure and raises the question of what do exclusive-or and DUE have mathematically in common?

Conclusion

Dynamic Unary Encoding introduces new choices of how binary data is represented and processed. DUE emerged out of efforts to discover innovative encoding methods. DUE has been applied as a bijective transform changing one binary data file into millions of different files. DUE was an important part of a surjective data encoder where many choices of what binary code would represent a source \( n \)-bit string were generated. As a tool of deconstruction or construction binary data is processed bit by bit. Utilized as a mathematical object, the forms a data can take is limited only by imagination because anything that can be done and undone to a data can be a part of some encoding algorithm utilizing DUE. As a virtual data set dynamic unary requires only one element in order to access all \( k \) elements of a cycle. That quality also extends to simple encryption of data by simply cycling a data in it’s orbit.

It is not difficult to envision applications of dynamic unary in categories such as the design of microprocessors, robotics, artificial intelligence, cryptography, logic and control and possibly physics. All in all there is more to DUE than has been done. More to discover about Dynamic Unary and it’s applications. The questions now are what needs to be done, how is it to be applied and what results will be realized.

On a personal note, like Gottfried Leibniz\(^6\) I too find the relationship of binary numbers and the I-Ching\(^7\) interesting. Dynamic Unary introduces a quality that I believe Leibniz would find interesting and that is, as noted in the I-Ching and Dynamic Unary Encoding, that one element can change to another. Metaphorically speaking, a Gate of Wonders\(^8\) has been opened and the garden that is DUE awaits.

\(^5\)Table One presents the encode data in left to right order. Decode direction equals right to left order.
\(^6\)Duyvendal’s translation is recommended as the context of this comment.
Acknowledgments

I wish to thank Gustavus Simmons for the paper Parity Encoding of Binary Sequences which has provided both an independent validation of the cycle structure and a guide for this my first paper.

Thanks goes to Henning Makholm and math.stackexchange.com for assistance with the element progression equation.

J.J.L. Duyvendal’s translation of chapter one of the Tao Te Ching is unique among many and has been an inspiration in my efforts to understand the nature of information.

Also, thanks goes to Mark Nelson whose Million Digit Challenge has provided me with a trustworthy data-set and an incredibly difficult challenge. And to all those who have been a part of that experience through the years.

Dedication

In memory of James Herbert Jones; my step-father not forgotten, observing his early Sunday morning maths doodles with coffee was the genesis of my realization that there is a realm of creativity to Mathematics.

References

[1] Khalid Sayood. Data Compression, 3rd ed, Morgan Kaufmann. ISBN-10: 012620862X — ISBN-13: 978-0126208627

[2] Binary number - Wikipedia, the free encyclopedia, Wikipedia.org

[3] Anderson, Marlow and Feil, Todd (2005), A First Course in Abstract Algebra, Chapman and Hall/CRC; 2nd edition. ISBN 1-58488-515-7.

[4] John Fraleigh, (2002), A first course in abstract algebra (7th ed.), Addison Wesley, ISBN 978-0-201-76390-4

[5] Gustavus J. Simmons. Parity Encoding of Binary Sequences. Designs, Codes and Cryptography, 27,157–164, 2002

[6] Eric J.Aiton. Leibniz: A Biography, Taylor and Francis. (1985) pp. 245–8. ISBN 0-85274-470-6.

[7] Edward Hacker; Steve Moore; Lorraine Patsco (2002). I Ching: An Annotated Bibliography. Routledge. ISBN 978-0-415-93969-0.

[8] J.J.L. Duyvendal, Tao Te Ching, Pg 17, ISBN 0 8048 1813 4

[9] Exclusive or - Wikipedia, the free encyclopedia, Wikipedia.org

[10] Marvin L. Bittinger, Logic, Proof and Sets, ISBN 0-201-10384-2
Encoding reads same parity bit segments and writes their lengths as unary code using the parity reference as the terminus parity of the unary code.

Figure 1: Example encode algorithm
Figure 2: Example decode algorithm

Decode must first determine the parity order so it reads each unary code until the current read bit position is equal to or greater than the parity reference position. Once the correct starting parity is known decode then reads unary symbols and writes parity segments.
Parity reference

Length of string 8 bits.

| Number of Cycles | Number of Elements |
|------------------|--------------------|
| 16               | 16                 |
| 32               | 8                  |

Parity reference by Number of Cycles of 16 Elements

Parity reference by Number of Cycles of 32 Elements

Parity reference by Number of Cycles of 8 Elements

---

**Table 1:**

Elements and cycles for string lengths one through eight bits in the encode direction.