Turing Machine to Compute Binary Carry Sequence

Kayla Brown, *Maryville College, Computer Science and Mathematics Double Major*

***Abstract*—The Binary Carry sequence is the sequence $a_n = \{a_{n-1}, n-1, a_{n-1}\}$ where $n \in \mathbb{N}$ and $a_1 = \{0\}$. Its application is to find the value of m for every n such that 2^m evenly divides n. The value of m can be found where n is the n^{th} position in the sequence and m is the corresponding number. Despite the fact that the Binary Carry sequence appears to be a relevant and valuable sequence, little research has been conducted towards the applications of the sequence to practical mathematical problems. We have created a Turing machine that accurately computes each value of the Binary Carry sequence. The Turing machine's general and main m-configurations are described in full detail in section II. Using the generated Turing machine as a basis for further exploration, we can anticipate future work to include the calculation of primes using the methods describes and additional similar methods, the detection of new patterns or sequences as a result of observing large value of n in the sequence, and the further research of the optimization of dissemination of information about traffic and road conditions using the Binary Carry sequence. We are optimistic that the development of this Turing machine will assist and promote further research on the topic and prompt new discoveries in the math world.**

I. Introduction

The Binary Carry sequence is a seemingly useful sequence in mathematics but has had little research and few known applications. The sequence itself and some of the known applications are described in the following subsections.

We have constructed a Turing machine, described in section II, that computes the infinite Binary Carry sequence for all values of $n \in \mathbb{N}$. The Turing machine will allow us to calculate the n^{th} position of the sequence at very large values of n.

Even though the Binary Carry sequence can be shown to be an applicable and useful sequence, little fieldwork has been conducted to see how we can apply it to practical mathematical problems. We hope that the construction of this Turing machine will assist and promote further research on the topic.

A. Binary Carry Sequence

For every natural number n, there exists a natural number m such that 2 to the power of m evenly divides n. There exists such a sequence, the Binary Carry sequence, that will indicate the value of m for every n. The Binary Carry sequence can be derived using the recursive equation $a_n = \{a_{n-1}, n-1, a_{n-1}\}$ where $n \in \mathbb{N}$ and $a_1 = \{0\}$ [1]. The first few derivations of the sequence are shown in the table below.

n	a_n
1	0
2	0 1 0
3	0 1 0 2 0 1 0
4	0 1 0 2 0 1 0 3 0 1 0 2 0 1 0
5	0 1 0 2 0 1 0 3 0 1 0 2 0 1 0 4 0 1 0 2 0 1 0 3 0 1 0 2 0 1 0

The value of m for every n such that $2^m|n$ can be found in the sequence where n is the n^{th} position in the sequence and m is the corresponding number. For example, to find the value of m for 8 where $2^m|8$, we will locate the 8^{th} position in the sequence of which the corresponding value is 3. Therefore, the highest power of 2 which evenly divides 8 is 3, hence $2^3|8$.

B. Applications

The main purpose of the Binary Carry sequence is to compute the highest power of 2 that evenly divides any natural number, but the sequence corresponds to many other solutions to problems in mathematics.

One of the more well-known problems that the sequence corresponds to is the solution to the Tower of Hanoi. The sequence directly corresponds to one less than the number of disks to be moved at the n^{th} step in the optimal solution to the Tower of Hanoi problem [1]. The Tower of Hanoi is a mathematical puzzle that consists of three rods, and any number of disks of different sizes which can slide onto any rod. Following a given set of rules, the solved when all disks are slid onto the last rod. It is left as a task to the reader to conduct more research on the Tower of Hanoi if desired. To visually show the correspondence of the two sequences, both of the sequences are shown below.

Binary Carry	{0 1 0 2 0 1 0 3 0 1 0 2 0 1 0 4 0 1 0 2 0 1 0 3 . . .}
Tower of Hanoi	{1 2 1 3 1 2 1 4 1 2 1 3 1 2 1 5 1 2 1 3 1 2 1 4 . . .}

For another solution that the sequence corresponds to, we must first find the parity of the Binary Carry sequence. The parity of an integer is its attribute of being odd or even. To find the parity of the sequence, we will replace each even integer, including 0, with 1 and each odd integer with 0. The Binary Carry sequence and its parity is shown below.

Binary Carry	{0 1 0 2 0 1 0 3 0 1 0 2 0 1 0 4 0 1 0 2 0 1 0 3 . . .}
Parity	{1 0 1 1 1 0 1 0 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 0 . . .}

The parity of the Binary Carry sequence corresponds to the accumulation point of 2^n cycles through successive bifurcations [1]. This problem will not be described in this paper but is left as a task to the reader if further information is desired.

In addition to the sequence corresponding to solutions to problems in mathematics, the sequence has been shown by an MIT study to optimize the dissemination of information about traffic and road conditions through networks of wirelessly connected cars.

The algorithm is used in modelling the ordering of data transmissions such that high-priority data is broadcasted more frequently but low-priority data is not excluded. In this model, lower numbers in the Binary Carry sequence, such as 0, will represent high-priority data, while the larger numbers in the sequence, such as 8, will represent low-priority data. We can see from this model that the lower numbers occur more frequently, hence the higher priority data is broadcasted more frequently, and the larger numbers occur less frequently, hence the lower priority data will still be broadcasted but not nearly as often [2].

For this application, the algorithm is being tested by T. J. Giuli, a technical expert on mobile computing at Ford Research and Advanced Engineering. According to MIT News, Guili says that "disseminating data through networks of cars will be particularly useful in urban areas" and "that networked vehicles could also help propagate traffic information in rural areas where cell towers – and, for that matter, hovering helicopters and DOT sensors – are sparse" [2].

II. Approach

The Turing machine that has been constructed to compute the Binary Carry sequence is described in the following subsections. We will begin by describing the alphabet, legend, and conventions of the Turing machine that will be used in the description of the *m*-functions and *m*-configurations. Next, we will describe the general *m*-functions and *m*-configurations that will be used in the main table of our Turing machine. Then, we will describe the main table and *m*-configurations of our Turing machine. Finally, a proof of correctness is constructed to prove that the machine works for any $n \in \mathbb{N} \ni n > 2$.

A. Alphabet, Legend, and Conventions

The Turing machine used to compute the Binary Carry sequence will be denoted as $\mathfrak{BC}$.

The tape of $\mathfrak{BC}$ uses the convention of alternating F-squares and E-squares. Both F-squares and E-squares are readable and writable where E-squares are used to mark the preceding F-squares. The leftmost point of the portion of the tape used in the computation is identified by the symbol ə placed in a pair of adjacent squares.

Because the following words and groups of words will be used repeatedly throughout the description of $\mathfrak{BC}$, we will abbreviate them with the indicated symbols. The legend of the symbols and their meaning found in the descriptions of *m*-functions and *m*-configurations of $\mathfrak{BC}$ is below.

Symbol	*Meaning*
E	Erase symbol from designated square
P	Print symbol to designated square
L	Move left one square
R	Move right one square
∀	Any symbol is read in the current square
None	No symbol is read in the current square
Σ	The longest sequence of consecutive 1s on the tape

The alphabet of this tape consists of the following symbols. The description of their use in the general case is also noted.

Symbol	*General Description*
ə	Marks the beginning of the tape
α	Marks the F-square to the left of the beginning of Σ
δ	Marks the last F-square in Σ
ω	Marks the beginning of the next Σ
ψ	Marks the first F-square to the left of the symbol ə
†	Marks the most recent 1 copied to the end of the tape
ϕ	Marks the most recent 0 copied to the end of the tape
0	Represents the number 0
1	Represents the number $\bar{m_0} = 1\ 1\ .\ .\ .\ 1$ of length $m_0 + 1$

Because $a_1 = \{0\}$ and Σ is of length zero for this sequence, we will let the initial state of the tape used in the computation of the Binary Carry sequence read

...	ə	ə	0	α	1	δ	0			...

which is the sequence of $a_2 = \{0\ 1\ 0\}$ such that α marks the F-square to the left of the beginning of Σ and δ marks the last F-square in Σ. We will let this be the case so that our *m*-configurations that require the detection of Σ do not have to consider the possibility that there does not exist any sequence of 1s. If desired, one can implement additional *m*-functions to initialize the tape in this manner.

B. General Purpose m-functions

The following *m*-functions are general functions. The parameters are to be passed through the function when it is called. In the following descriptions, α and δ represent symbols that are passed through the function and β represents *m*-functions that are passed through the function. Each table begins with the name of the function and its parameters, then, based on the symbol that is read on the current square, the function will then perform the designated operations and move onto the final *m*-configuration.

$l(\beta)$**: move left** The *m*-function performs the basic operation L then moves to *m*-configuration β.

m-config.	*Symbol*	*Operations*	*Final m-config.*
$l(\beta)$		L	β

$r(\beta)$**: move right** The *m*-function performs the basic operation R then moves to *m*-configuration β.

m-config.	*Symbol*	*Operations*	*Final m-config.*
$r(\beta)$		R	β

$rp(\alpha, \beta)$**: replace** The *m*-function replaces the current symbol with the symbol α then moves to *m*-configuration β.

m-config.	*Symbol*	*Operations*	*Final m-config.*
$rp(\alpha, \beta)$		Pα	β

$mc(\alpha, \beta)$**: mark** The *m*-function marks the current F-square by printing the symbol α in the adjacent E-square then moves to *m*-configuration β.

m-config.	*Symbol*	*Operations*	*Final m-config.*
$mc(\alpha, \beta)$		RPα	β

$rm(\alpha, \beta)$**: erase** The *m*-function erases the leftmost occurrence on the tape of the symbol α then moves to *m*-configuration β.

m-config.	*Symbol*	*Operations*	*Final m-config.*
$rm(\alpha, \beta)$			$f(\alpha, rm_1(\beta))$
$rm_1(\beta)$		E	β

$pe(\alpha, \beta)$**: print at end** The *m*-function prints the symbol α at the end of the sequence of symbols on the tape then moves to *m*-configuration β.

m-config.	*Symbol*	*Operations*	*Final m-config.*
$pe(\alpha, \beta)$			$f(ə, pe_1(\alpha, \beta))$
$pe_1(\alpha, \beta)$	$\forall$	RR	$pe_1(\alpha, \beta)$
	None	Pα	β

$pem(\alpha, \delta, \beta)$**: print at end and mark** The *m*-function prints the symbol α on the F-square at the end of the sequence of symbols on the tape and marks it with the symbol δ then moves to *m*-configuration β.

m-config.	*Symbol*	*Operations*	*Final m-config.*
$pem(\alpha, \delta, \beta)$			$pe(\alpha, mc(\delta, \beta))$

$f(\alpha, \beta)$**: find** The *m*-function finds the leftmost occurrence on the tape of the symbol α then moves to *m*-configuration β.

m-config.	*Symbol*	*Operations*	*Final m-config.*
$f(\alpha, \beta)$	ə	L	$f_1(\alpha, \beta)$
	$\bar{ə}$	L	$f(\alpha, \beta)$
$f_1(\alpha, \beta)$	α		β
	$\bar{\alpha}$	R	$f_1(\alpha, \beta)$

$f'(\alpha, \beta)$**: find and move left** The *m*-function find the leftmost occurrence on the tape of the symbol α, moves one square to the left, then moves to *m*-configuration β.

m-config.	*Symbol*	*Operations*	*Final m-config.*
$f'(\alpha, \beta)$			$f(\alpha, l(\beta))$

$f''(\alpha, \beta)$**: find and move right** The *m*-function find the leftmost occurrence on the tape of the symbol α, moves one square to the right, then moves to *m*-configuration β.

m-config.	*Symbol*	*Operations*	*Final m-config.*
$f''(\alpha, \beta)$			$f(\alpha, r(\beta))$

$fr(\alpha, \delta, \beta)$**: find and replace** The *m*-function find the leftmost occurrence on the tape of the symbol α, replaces the current symbol with the symbol δ, then moves to *m*-configuration β.

m-config.	*Symbol*	*Operations*	*Final m-config.*
$fr(\alpha, \delta, \beta)$			$f(\alpha, rp(\delta, \beta))$

$eof(\beta)$**: find end of tape** The *m*-function moves to the start of the tape, then looks for the first F-square with no symbol then moves to *m*-configuration β.

m-config.	*Symbol*	*Operations*	*Final m-config.*
$eof(\beta)$			$f(ə, eof_1)$
$eof_1(\beta)$	$\forall$	RR	$eof(\beta)$
	None		β

$c(\alpha, \delta, \beta)$**: copy** The *m*-function copies the symbols in the F-squares between the symbols α and δ to the end of the tape then moves to *m*-configuration β.

m-config.	*Symbol*	*Operations*	*Final m-config.*
$c(\alpha, \delta, \beta)$			$f'(\alpha, c_1(\alpha, \delta, \beta))$
$c_1(\alpha, \delta, \beta)$	0	R	$c_2(\alpha, \delta, \beta)$
	1	R	$c_3(\alpha, \delta, \beta)$
$c_2(\alpha, \delta, \beta)$	ϕ	ER	$c_1(\alpha, \delta, \beta)$
	δ	Pϕ	$pe(0, rm(\phi, \beta))$
	α	ER	$c_1(\alpha, \delta, \beta)$
	None	Pϕ	$pe(0, c(\phi, \delta, \beta))$
$c_3(\alpha, \delta, \beta)$	$\dagger$	ER	$c_1(\alpha, \delta, \beta)$
	δ	P$\dagger$	$pe(1, rm(\dagger, \beta))$
	α	ER	$c_1(\alpha, \delta, \beta)$
	None	P$\dagger$	$pe(1, c(\dagger, \delta, \beta))$

C. Main Table

The following *m*-configurations describe the process of 𝔅ℭ. The symbols in these tables are not arbitrary. 𝔅ℭ calls the *m*-configuration mid first then continues to call the indicated *m*-functions and will result in an infinite recursion. Because the first *m*-function mid will find the end of the tape regardless of the position of the current square, we can say that that 𝔅ℭ will start with the current square on the F-square with the symbol ə.

mid**: Mark End Configuration** The *m*-configuration mid marks the end of the tape with the symbol ω then moves to *m*-configuration lnc. This configuration therefore marks the beginning of the next longest sequence of 1s because lnc will place this sequence here.

m-config.	*Symbol*	*Operations*	*Final m-config.*
mid			$eof(mid_1)$
mid_1		LPω	lnc

lnc: **Print Next Σ** The *m*-configuration lnc copies the current Σ to the end of the tape. lnc then replaces the symbol ω with the symbol α to indicate the beginning of the next Σ. Finally, lnc prints the symbol 1 marked with the symbol δ then the symbol 0 to the end of the tape and moves to *m*-configuration ca. Now, the current Σ is between the symbols α and δ.

m-config.	*Final m-config.*
lnc	$c(\alpha, \delta, lnc_1)$
lnc_1	$fr(\omega, \alpha, lnc_2)$
lnc_2	$pem(1, \delta, pe(0, ca))$

ca: **Copy sequence preceding Σ to end of tape** The *m*-configuration ca marks the beginning of the tape with the symbol ψ then copies the sequuence between the symbols ψ and α, which is the sequence preceding Σ, to the end of the tape. Doing this results in the symbol α being erased, so ca then moves to the *m*-configuration mb.

m-config.	*Final m-config.*
ca	$f''(ə, r(mc(\psi, ca_2)))$
ca_2	$c(\psi, \alpha, mb)$

mb: **mark Σ** The *m*-configuration mb finds the F-square marked with the symbol δ and moves left over the preceding F-squares until the sequence of 1s terminates, indicated by a 0 in the current F-square, then marks this F-square with the symbol α, then moves to *m*-configuration mid.

m-config.	*Symbol*	*Operations*	*Final m-config.*
mb			$f'(\delta, mb_1)$
mb_1	1	LL	mb_1
	0		$mc(\alpha, mid)$

D. Proof of Correctness

Let $n \in \mathbb{N} \mid n > 2$ and $a_n = \{a_{n-1}, n-1, a_{n-1}\}$ where $\{n+m\} \mid m \in \mathbb{Z}$ represents the longest sequence of consecutive 1s. Let $\mathfrak{BC}$ be the Turing machine described in the previous subsections. If we observe the state of the Turing machine succeeding *m*-configuration ca_2 and preceding the *m*-configuration mib, it is clear that the state describes the sequence of $a_{n-1} = \{a_{n-2}, n-2, a_{n-2}\}$.

First consider $n = 3$. By observation, the tape preceding the *m*-configuration mib of computing $n = 3$ reads $\{a_2\} = \{0, 1, 0\}$. $\mathfrak{BC}$ finds the Σ in $\{a_2\}$, which is $\{1\}$, and copies this sequence to the end of our tape then prints an additional 1. Now, the tape reads $\{a_2, 1, 1\} = \{a_2, 2\}$.

Next, $\mathfrak{BC}$ copies from the beginning of the tape to the beginning of $\{2\}$ to the end of our tape. Now the tape reads $\{a_2, 2, a_2\}$, which by definition is equal to the sequence a_3. Hence, $\mathfrak{BC}$ successfully computes a_n for $n = 3$.

Now we consider the sequence a_n of n for any $n > 3$. $\mathfrak{BC}$ finds the Σ in $\{a_{n-1}\}$, which is $\{n-2\}$, and copies this sequence to the end of our tape then prints an additional 1. Now, the tape reads $\{a_{n-1}, n-1\}$.

Next, $\mathfrak{BC}$ copies from the beginning of the tape to the beginning of $\{n-1\}$ to the end of our tape. Now the tape reads $\{a_{n-1}, n-1, a_{n-1}\}$, which is equal to the sequence of a_n by definition. Hence, $\mathfrak{BC}$ successfully computes a_n of n.

Now consider $n+1$. $\mathfrak{BC}$ repeats the process as before, first by finding the Σ in $\{a_n\}$, which is $\{n-1\}$, and copies it to the end of our tape followed by printing an additional 1. Now our tape reads $\{a_n, n\}$.

As before, $\mathfrak{BC}$ then copies from the beginning of our tape to the beginning of $\{a_n\}$ to the end of our tape. Now the tape reads $\{a_n, n, a_n\}$, which is equal to the sequence of a_{n+1} by definition. Hence, $\mathfrak{BC}$ successfully computes a_{n+1} of $n+1$.

Therefore, by induction, $\mathfrak{BC}$ successfully computes the sequence a_n for $\forall\ n \in \mathbb{N} \mid n > 2$ and $a_n = \{a_{n-1}, n-1, a_{n-1}\}$. ■

III. Conclusion

We have constructed a Turing machine that we have proved to successfully compute the Binary Carry sequence for any $n \in \mathbb{N} \mid n > 2$.

Further research on the topic can include continuing testing the use of the sequence to optimize the dissemination of information about traffic and road conditions through networks of wirelessly connected cars.

Future work on the topic can include implementing the Turing machine to calculate significantly large values of n. Perhaps a pattern different from $a_n = \{a_{n-1}, n-1, a_{n-1}\}$ can be detected when the values of n begin to approach numbers that are large enough to be referred to as infinity.

Another possibility for future work is to determine primes using the Binary Carry sequence and sequences like it. The Binary Carry sequence is used to compute the value of $m \in \mathbb{N}$ for every $n \in \mathbb{N}$ such that $2^m | n$ where n is the n^{th} position in the sequence and m is the corresponding number. The first step to computing primes using this approach is to discover sequences that indicate the value of $m \in \mathbb{N}$ for every $n \in \mathbb{N}$ such that $3^m|n$, $4^m|n$, $5^m|n$, and so on. The next step would be to create a Turing machine to compute the values of each position in the sequence for every sequence we have discovered. Then, for the i^{th} position in any sequence, a Turing machine exists to calculate the sequence that indicates the value of m for $i^m|n$. When all the Turing machines are ran simultaneously, we are able to observe every natural divisor of every natural number. For any ith position in any sequence, say we observe that every corresponding m value is 0 except for in the sequence which indicates the value of m for $i^m|n$ such that this value is 1. Hence, the number i has only the divisor of itself and 1, therefore i is prime.

While the Binary Carry sequence is shown to be an applicable and useful sequence, little research has been conducted to see how we can apply it to practical mathematical problems. We are optimistic that the construction of this Turing machine will assist and promote further research on the topic as described and lead to new discoveries in the math world.

REFERENCES

[1] E. Weisstein, *Binary Carry Sequence*, MathWorld – A Wolfram Web Resource, 2015. [Online]. Available: http://mathworld.wolfram.com/BinaryCarrySequence.html. [Accessed: 01 Nov 2015].

[2] L. Hardesty, *Cars as traffic sensors*, MIT News, 2010. [Online]. Available: http://news.mit.edu/2010/cars-sensors-0924. [Accessed: 02 Nov 2015].