History of the numeral systems essay sample

A numeral system (or system of numeration) is a linguistic system and mathematical notation for representing numbers of a given set by symbols in a consistent manner. For example, It allows the numeral “ 11” to be interpreted as the binary numeral for three, the decimal numeral for eleven, or other numbers in different bases. Ideally, a numeral system will: • Represent a useful set of numbers (e. g. all whole numbers, integers, or real numbers) • Give every number represented a unique representation (or at least a standard representation) • Reflect the algebraic and arithmetic structure of the numbers. For example, the usual decimal representation of whole numbers gives every whole number a unique representation as a finite sequence of digits, with the operations of arithmetic (addition, subtraction, multiplication and division) being present as the standard algorithms of arithmetic.

However, when decimal representation is used for the rational or real numbers, the representation is no longer unique: many rational numbers have two numerals, a standard one that terminates, such as 2. 31, and another that recurs, such as 2. 309999999… . Numerals which terminate have no non-zero digits after a given position. For example, numerals like 2. 31 and 2. 310 are taken to be the same, except in the experimental sciences, where greater precision is denoted by the trailing zero. The most commonly used system of numerals is known as Hindu-Arabic numerals. Great Indian mathematicians Aryabhatta of Kusumapura (5th Century) developed the place value notation. Brahmagupta (6th Century) introduced the symbol zero. Unary System: Every natural number is represented by a corresponding number of symbols, for example the number seven would be represented by ///////.

Elias gamma coding which is commonly used in data compression expresses arbitrary-sized numbers by using unary to indicate the length of a binary numeral. With different symbols for certain new values, if / stands for one, – for ten and + for 100, then the number 123 as + – – /// without any need for zero. This is called signvalue notation. More elegant is a positional system, also known as place-value notation. Again working in base 10, we use ten different digits 0, …, 9 and use the position of a digit to signify the power of ten that the digit is to be multiplied with, as in 304 = 3×100 + 0×10 + 4×1. Note that zero, which is not needed in the other systems, is of crucial importance here, in order to be able to " skip" a power. In certain areas of computer science, a modified base-k positional system is used, called bijective numeration, with digits 1, 2, …, k (k ≥ 1), and zero being represented by the empty string. This establishes a bijection between the set of all such digit-strings and the set of non-negative integers, avoiding the non-uniqueness caused by leading zeros. Bijective base-k numeration is also called k-adic notation, not to be confused with padic numbers. Bijective base-1 the same as unary.

Five A base-5 system (quinary), on the number of fingers, has been used in many cultures for counting. It may also be regarded as a sub-base of other bases, such as base 10 and base 60. Eight A base-8 system (octal), spaces between the fingers , was devised by the Yuki of Northern California, who used the spaces between the fingers to count, corresponding to the digits one through eight Ten The base-10 system (decimal) is the one most commonly used today. It is assumed to have originated because humans have ten fingers. These systems often use a larger superimposed base. Twelve Base-12 systems (duodecimal or dozenal) have been popular. It is the smallest multiple of one, two, three, four and six. There is still a special word for 121, dozen and a word for 122, gross. Multiples of 12 have been in common use as English units of resolution in the analog and digital printing world, where 1 point equals 1/72 of an inch and 12 points equal 1 pica, and printer resolutions like 360, 600, 720, 1200 or 1440 dpi (dots per inch) are common. These are combinations of base-12 and base-10 factors: (3×12)×10, (5×12)×10, (6×12)×10, (10×12)×10 and (12×12)×10.

Twenty The Maya civilization and other civilizations of Pre-Columbian Mesoamerica used base-20 (vigesimal). Remnants of a Gaulish base-20 system also exist in French, as seen today in the names of the numbers from 60 through 99. The Irish language also used base-20 in the past. Danish numerals display a similar base20 structure. Sixty Base 60 (sexagesimal) was used by the Sumerians and their successors in Mesopotamia and survives today in our system of time (hence the division of an hour into 60 minutes and a minute into 60 seconds) and in our system of angular measure (a degree is divided into 60 minutes and a minute is divided into 60 seconds). 60 also has a large number of factors, including the first six counting numbers.

Base-60 systems are believed to have originated through the merging of base-10 and base-12 systems. Dual base (five and twenty) Many ancient counting systems use 5 as a primary base, almost surely coming from the number of fingers on a person’s hand. Often these systems are supplemented with a secondary base, sometimes ten, sometimes twenty. In some African languages the word for 5 is the same as “ hand” or “ fist”. Counting continues by adding 1, 2, 3, or 4 to combinations of 5, until the secondary base is reached. In the case of twenty, this word often means “ man complete”. This system is referred to as quinquavigesimal. It is found in many languages of the Sudan region.

BINARY The ancient Indian writer Pingala developed advanced mathematical concepts for describing prosody, and in doing so presented the first known description of a binary numeral system. A full set of 8 trigrams and 64 hexagrams, analogous to the 3-bit and 6-bit binary numerals, were known to the ancient Chinese in the classic text I Ching. An arrangement of the hexagrams of the I Ching, ordered according to the values of the corresponding binary numbers (from 0 to 63), and a method for generating the same, was developed by the Chinese scholar and philosopher Shao Yong in the 11th century. In 1854, British mathematician George Boole published a landmark paper detailing an algebraic system of logic that would become known as Boolean algebra.

His logical calculus was to become instrumental in the design of digital electronic circuitry. In 1937, Claude Shannon produced his master’s thesis at MIT that implemented Boolean algebra and binary arithmetic using electronic relays and switches for the first time in history. Entitled A Symbolic Analysis of Relay and Switching Circuits, Shannon’s thesis essentially founded practical digital circuit design. In November 1937, George Stibitz, then working at Bell Labs, completed a relay-based computer he dubbed the “ Model K” (for “ Kitchen”, where he had assembled it), which calculated using binary addition. The Complex Number Computer was completed by January 8, 1940, was able to calculate complex numbers. On September 11, 1940, Stibitz was able to send the Complex Number Calculator remote commands over telephone lines by a teletype.

Reflective Code A code is said to be reflective when code for 9 is complement for the code for 0, and so is for 8 and 1 codes, 7 and 2, 6 and 3, 5 and 4. Codes 2421, 5211, and excess-3 are reflective, whereas the 8421 code is not. Sequential Codes

A code is said to be sequential when two subsequent codes, seen as numbers in binary representation, differ by one. This greatly aids mathematical manipulation of data. The 8421 and Excess-3 codes are sequential, whereas the 2421 and 5211 codes are not. Non weighted codes

Non weighted codes are codes that are not positionally weighted. That is, each position within the binary number is not assigned a fixed value. Ex: Excess-3 code Excess-3 Code

Excess-3 is a non weighted code used to express decimal numbers. The code derives its name from the fact that each binary code is the corresponding 8421 code plus 0011(3). Gray Code The gray code belongs to a class of codes called minimum change codes, in which only one bit in the code changes when moving from one code to the next. The Gray code is non-weighted code, as the position of bit does not contain any weight. The gray code is a reflective digital code which has the special property that any two subsequent numbers codes differ by only one bit. This is also called a unit-distance code. In digital Gray code has got a special place.

Gray Code MSB is binary code MSB. Gray Code MSB-1 is the XOR of binary code MSB and MSB-1. MSB-2 bit of gray code is XOR of MSB-1 and MSB-2 bit of binary code. MSB-N bit of gray code is XOR of MSB-N-1 and MSB-N bit of binary code.

Error-Detection Codes Binary information may be transmitted through some communication medium, e. g. using wires or wireless media. A corrupted bit will have its value changed from 0 to 1 or vice versa. To be able to detect errors at the receiver end, the sender sends an extra bit (parity bit) with the original binary message.

A parity bit is an extra bit included with the n-bit binary message to make the total number of 1’s in this message (including the parity bit) either odd or even. If the parity bit makes the total number of 1’s an odd (even) number, it is called odd (even) parity. The table shows the required odd (even) parity for a 3-bit message. At the receiver end, an error is detected if the message does not match have the proper parity (odd/even). Parity bits can detect the occurrence 1, 3, 5 or any odd number of errors in the transmitted message. At the receiver end, an error is detected if the message does not match have the proper parity (odd/even). Parity bits can detect the occurrence 1, 3, 5 or any odd number of errors in the transmitted message. No error is detectable if the transmitted message has 2 bits in error since the total number of 1’s will remain even (or odd) as in the original message. In general, a transmitted message with even number of errors cannot be detected by the parity bit.

Binary information may be transmitted through some communication medium, e. g. using wires or wireless media. Noise in the transmission medium may cause the transmitted binary message to be corrupted by changing a bit from 0 to 1 or vice versa. To be able to detect errors at the receiver end, the sender sends an extra bit (parity bit). Gray Code

The Gray code consists of 16 4-bit code words to represent the decimal Numbers 0 to 15. For Gray code, successive code words differ by only one bit from one to the next as shown in the table and further illustrated in the Figure.

Error detection codes The general idea for achieving error detection and correction is to add some redundancy (i. e., some extra data) to a message, which receivers can use to check consistency of the delivered message, and to recover data determined to be erroneous. Error-detection and correction schemes can be either systematic or nonsystematic: In a systematic scheme, the transmitter sends the original data, and attaches a fixed number of check bits (or parity data), which are derived from the data bits by some deterministic algorithm. If only error detection is required, a receiver can simply apply the same algorithm to the received data bits and compare its output with the received check bits; if the values do not match, an error has occurred at some point during the transmission. In a system that uses a non-systematic code, the original message is transformed into an encoded message that has at least as many bits as the original message. Good error control performance requires the scheme to be selected based on the characteristics of the communication channel. Common channel models include memory-less models where errors occur randomly and with a certain probability, and dynamic models where errors occur primarily in bursts.

Consequently, error-detecting and correcting codes can be generally distinguished between random-errordetecting/correcting and burst-error-detecting/correcting. Some codes can also be suitable for a mixture of random errors and burst errors. If the channel capacity cannot be determined, or is highly varying, an error-detection scheme may be combined with a system for retransmissions of erroneous data. This is known as automatic repeat request (ARQ), and is most notably used in the Internet. An alternate approach for error control is hybrid automatic repeat request (HARQ), which is a combination of ARQ and error-correction coding. Error detection schemes Error detection is most commonly realized using a suitable hash function (or checksum algorithm). A hash function adds a fixed-length tag to a message, which enables receivers to verify the delivered message by recomputing the tag and comparing it with the one provided. There exists a vast variety of different hash function designs.

However, some are of particularly widespread use because of either their simplicity or their suitability for detecting certain kinds of errors (e. g., the cyclic redundancy check’s performance in detecting burst errors). Random-error-correcting codes based on minimum distance coding can provide a suitable alternative to hash functions when a strict guarantee on the minimum number of errors to be detected is desired. Repetition codes, described below, are special cases of error-correcting codes: although rather inefficient, they find applications for both error correction and detection due to their simplicity. Repetition codes A repetition code is a coding scheme that repeats the bits across a channel to achieve error-free communication. Given a stream of data to be transmitted, the data is divided into blocks of bits. Each block is transmitted some predetermined number of times. For example, to send the bit pattern “ 1011”, the four-bit block can be repeated three times, thus producing “ 1011 1011 1011”. However, if this twelve-bit pattern was received as “ 1010 1011 1011” – where the first block is unlike the other two – it can be determined that an error has occurred.

Repetition codes are not very efficient, and can be susceptible to problems if the error occurs in exactly the same place for each group (e. g., “ 1010 1010 1010” in the previous example would be detected as correct). The advantage of repetition codes is that they are extremely simple, and are in fact used in some transmissions of numbers stations.

Parity bits A parity bit is a bit that is added to a group of source bits to ensure that the number of set bits (i. e., bits with value 1) in the outcome is even or odd. It is a very simple scheme that can be used to detect single or any other odd number (i. e., three, five, etc.) of errors in the output. An even number of flipped bits will make the parity bit appear correct even though the data is erroneous. Extensions and variations on the parity bit mechanism are horizontal redundancy checks, vertical redundancy checks, and “ double,” “ dual,” or “ diagonal” parity (used in RAID-DP). Checksums A checksum of a message is a modular arithmetic sum of message code words of a fixed word length (e. g., byte values). The sum may be negated by means of a one’s-complement prior to transmission to detect errors resulting in all-zero messages. Checksum schemes include parity bits, check digits, and longitudinal redundancy checks. Some checksum schemes, such as the

Luhn algorithm and the Verhoeff algorithm, are specifically designed to detect errors commonly introduced by humans in writing down or remembering identification numbers.

Cyclic redundancy checks (CRCs)

A cyclic redundancy check (CRC) is a single-burst-error-detecting cyclic code and non-secure hash function designed to detect accidental changes to digital data in computer networks. It is characterized by specification of a so-called generator polynomial, which is used as the divisor in a polynomial long division over a finite field, taking the input data as the dividend, and where the remainder becomes the result. Cyclic codes have favorable properties in that they are well suited for detecting burst errors. CRCs are particularly easy to implement in hardware, and are therefore commonly used in digital networks and storage devices such as hard disk drives. Even parity is a special case of a cyclic redundancy check, where the single-bit CRC is generated by the divisor x+1. Cryptographic hash functions A cryptographic hash function can provide strong assurances about data integrity, provided that changes of the data are only accidental (i. e., due to transmission errors). Any modification to the data will likely be detected through a mismatching hash value. Furthermore, given some hash value, it is infeasible to find some input data (other than the one given) that will yield the same hash value. Message authentication codes, also called keyed cryptographic hash functions, provide additional protection against intentional modification by an attacker.

Error-correcting codes Any error-correcting code can be used for error detection. A code with minimum Hamming distance, d, can detect up to d-1 errors in a code word. Using minimum-distance-based error-correcting codes for error detection can be suitable if a strict limit on the minimum number of errors to be detected is desired. Codes with minimum Hamming distance d= 2 are degenerate cases of error-correcting codes, and can be used to detect single errors. The parity bit is an example of a single-error-detecting code. The Berger code is an early example of a unidirectional error(-correcting) code that can detect any number of errors on an asymmetric channel, provided that only transitions of cleared bits to set bits or set bits to cleared bits can occur.

Hamming Codes It is an error correction code that separates the bits holding the original value (data bits) from the error correction bits (check bits), and the difference between the calculated and actual error correction bits is the position of the bit that’s wrong. Error correction codes are a way to represent a set of symbols so that if any 1 bit of the representation is accidentally flipped, you can still tell which symbol it was. For example, you can represent two symbols x and y in 3 bits with the values x= 111 and y= 000. If you flip any one of the bits of these values, you can still tell which symbol was intended. If more than 1 bit changes, you can’t tell, and you probably get the wrong answer. So it goes; 1-bit error correction codes can only correct 1-bit changes. If b bits are used to represent the symbols, then each symbol will own 1+b values: the value representing the symbol, and the values differing from it by 1 bit. In the 3-bit example above, y owned 1+3 values: 000, 001, 010, and 100. Representing n symbols in b bits will consume n*(1+b) values. If there is a 1-bit error correction code of b bits for n symbols, then n*(1+b)