Article

Not peer-reviewed version

Philosophical and Mathematical Foundations of Any Number Generating System and The Lu-Number System

Joseph Willrich Lutalo *

Posted Date: 10 June 2025

doi: 10.20944/preprints202506.0790.v1

Keywords: Philosophy; Foundations; Information; Physics; Mensuration; Encoding; Processing; Numbers; Bases; Arithmetic; Operations; Symbol Sets; Identity; Quantity; Meaning

 doi:10.20944/preprints202506.0790.v1

Article

Philosophical and Mathematical Foundations of Any Number Generating System and The Lu-Number System

Joseph Willrich Lutalo

An Independent Researcher at Nuchwezi Research; jwl@nuchwezi.com, joewillrich@gmail.com

Abstract: This work sets out to theoretically construct a platform for allowing for the systematic generation of numbers (in any base) from physical (natural or artificial) or non-physical (virtual, conceptual or artificial for example) entropy sources — ultimately, the purpose is to support the generation of numbers conforming to some preferable distribution, especially the uniform distribution, and thus random numbers. The core of this platform is based on a means of reading raw information from the source of entropy, and then encoding it using an abstract number system we call the Lu-Number System (LNS). Then, starting from that basic LNS number generator — which shall produce numbers in the LNS base — *lu-Base*, more sophisticated generators can then be built atop that — like those meant to generate *higher-order numbers* or numbers in common bases such as binary, decimal or base-36. We can then be able to leverage LNS to implement more useful generators such as Random Bit Generators (RBGs), True Random Number Generators (TRNGs) as well as Cryptographically Secure RNGs (CSRNGs) among others. At the moment, this is the condensed edition of that work; thus prepared so it can readily be applicable/usable in theoretical and practical contexts across mathematics, philosophy, but also in various sciences and mathematical disciplines — especially in Computer Science, its originally intended audience[1].

Keywords: philosophy; foundations; information; physics; mensuration; encoding; processing; numbers; bases; arithmetic; operations; symbol sets; identity; quantity; meaning

1. A Philosophical Introduction

Assume we begin with the following definition:

[1] This research builds on earlier work in number theory by Joseph W. Lutalo started during his early explorations in graduate school — circa 2020, and whose first results culminated in the GTNC paper of 2024[1], and which has catalyzed further explorations in-relation to that paper since then. Besides earlier work by the author in Software Engineering, Software Language Engineering, Philosophy and Mathematics (**refer to https://bit.ly/profjwl**) especially as part of his formal graduate school load while at Makerere University, the present work is meant to provide support towards qualification for and/or contribute to the essential load required for a DPhil in Computer Science of Oxford University in case they accept this original thesis.

 doi:10.20944/preprints202506.0790.v1

Definition 1 (**Defining Θ: The Canonical Form Expression of any number ρ in [Any] Base β**). *The number ρ, in any base β shall be understood to mean, the expression Θ, that encodes the following identifying information concerning the number;*

1. *The **Quantity of** ρ: which, mathematically we shall express as $|\rho|_\beta$ to imply, the numeric size of the number when expressed in the base β.*
2. *The **Identity of** ρ: which, symbolically allows for the distinct naming of the number ρ - in any base first of all, but also in a particular base such as β. This, so that if any two numbers in base β, say ρ_β and τ_β, are such that Θ correctly identifies either of them, then $\rho_\beta = \tau_\beta$, otherwise they are distinct, and different quantities or identities.*

*Assuming the expression Θ correctly allows for the inference of the quantity of ρ in any base, as well as allows for the distinct identification and expression of ρ in that base, then it is a number — the **Canonical Form of** ρ, and which both identifies and quantifies the number ρ in any base β.*
An example expression for a number ρ_β expressible using Θ could be written as:

$$\Theta = \sum_{i=0}^{n\in\mathbb{N}} a_i = \sum_{i=0}^{n\in\mathbb{N}} k\beta^{n-i}$$

where $k \in \psi_\beta$ and $\beta \in \mathbb{N}$

By **Definition 1**, assuming we go by popular conventions in use for ordinary human language and semantics, then, when one entity declares to another that "I have counted 1000000", it could be interpreted as meaning or implying one of the following:

- '1000000' is a quantity.
- '1000000' also expresses the name of that quantity — its unique or distinct identifier in some base — typically, base-10 — and thus just "1 Million", unless explicitly specified; such as if it had been expressed as 1000000_2 or $0x1000000$ or perhaps $0b1000000$ — notations more commonly found in *computational literature*, and then $\overline{M}$ in Latin Numeral Notation.
- '1000000' implies it is possibly *just* the quantity, and nothing about whether or not the number has some non-quantifiable quality — such as whether it is *signed* or not.

The above information might be sufficient for the purposes of merely operating on the name and quantity of the number **in the usual or commonplace sense**, but perhaps not in a totally meaningful or rather, non-ambiguous way. What this means exactly, can be appreciated by further considering that:

1. Without knowing explicitly in which base the **counting operation** was conducted — or rather, how the computation was performed, we can only guess at what really the quantity of the expression '1000000' is. This, especially if we wish to somewhat reproduce or rather compare other quantities in some or any base, relative to that expressed quantity. So basically, without the base information, the expression '1000000' is but a *scalar symbol* — it tells us about some number and offers an idea about its quantity or size, but **actually** *only its name* and little or nothing about how to **exactly** express its quantity meaningfully.
2. As a result of the above condition, we might for example wrongfully mis-read the expression '1000000' to mean any of the following:
   - the decimal number 1000000 and a quantity as is the result of counting the legs of 10,000 centipedes.
   - especially because by **Theorem 2 (Membership in base-β)** in [2], the expression qualifies to be considered a base-2 number, and thus, makes the decimal quantity equivalent to the **Canonical Form** of the number it expresses (see **Definition 1**) to be the decimal number 64 — or rather, 64_{10} — refer to **Theorem 1 (Any Base to Decimal)** in [3], and which, by **Theorem 2 (Membership in base-β)** in [2] as well as **Theorem 2 (Defining ψ_{36})** in [3], also qualifies to be a number in base-36, by which case then, using the standard base conversion mechanics — like from one base into decimal and from decimal to another [4], and what we know of

 doi:10.20944/preprints202506.0790.v1

base-36[3], that the number expression '1000000' then is equivalent to $1S_{36}$. That is if we have treated the number as a base-10 expression and then converted it to base-36. However, if we indeed treat that same expression, '1000000', as a base-36 number, it then turns out to be equivalent to the decimal quantity 36^6 or rather, 2176782336_{10} and other interesting numbers in other bases...

Further, we might not clearly know whether the number is positive or not, or whether it expresses a pure number or a pure fractional number — since, for *pure fractionals* [1], we might, under some special notations, do away with the implied usual "symbolic" parts of the number's expression such as when we write "0.5" to mean $\frac{1}{2}$ when we could as well have written just ".5" or even perhaps 2^{-1}!

So, much as sometimes number expressions are used/meant to communicate *more precise*, or rather, *unambiguous* forms of quantities and quantifiably identifiable measures or properties of things, then, it is important that we be more careful and adopt more explicit, or rather rigorously unambiguous expressions of numbers when communicating exact science, mathematics and/or verifiable information.

A more precise way to report the result of a computation such as counting, should have been expressed as: "I have counted 1000000 in base 2."

2. The Lu-Number System

In this section, we are going to build upon ideas from the previous section, to try and develop from scratch, a theory and mechanics for defining abtract machines that can process information as input and then produce or rather, generate other information — in particular, information expressions such as numbers in some number system. These machines are essentially **Number Generators**, though, ultimately, our objective is to systematize and formalize and advance the important field of **Random Number Generators.**

We shall refer to the original reference number system we are going to base our number generation theory on, as the **Lu-Number System**, LNS.

2.1. LNS Foundations

Let there be a distinct symbol to be referred to as the signal. For practical purposes, let the signal be the symbol "<".

Axiom 1. *Every distinct information has a definite distinct way it can be expressed or encoded so as to express, manifest or represent that for which it is a reference—typically via some symbol or systematic sequence of them, which is known as "information encoding". Equivalently, symbols represent or express some definite information — information that can be understood and is communicable.*

Definition 2 (**The First Base Symbol: *the Signal***). *This is essentially the **Primal Signal Encoding Method** in the Lu-Number System (LNS); the primal-symbol, also to be known as "the signal", to be simply written as "<" — a single opening angle-bracket. In the LNS, it is meant to basically express the presence of some information — any perceptible information.*

Axiom 2. *In being expressed, any distinct information under the LNS can take on one of two forms; the primal-form or the alternate/inverted form, but not both simultaneously.*

Definition 3 (**Alternate Base-Symbol: *the Alternate Signal* — the *Anti-Signal***). *In LNS, because it is the alternate or opposite of signal, ">", it shall simply be expressed as ">" — the lateral inversion or exact mirror reflection of the primal-symbol. Also, this operation of reflection or inversion of symbols or information in general, is among the first permissible operations in LNS. More about this later.*

 doi:10.20944/preprints202506.0790.v1

Theorem 1 (The LNS Base Symbol Set: ψ_{LNS}). *The Lu-Number System has only two fundamental information-encoding symbols (also ones we might think of as the LNS numbers);*

- $<$ $\Longrightarrow$ *the* ***signal****.*
- $>$ $\Longrightarrow$ *the* ***anti-signal****.*

And based on what we know as of this moment, their order isn't fixed or well-defined yet — however, since the presence of information is greater than or more preferable than its absence, we might quantify signal to be greater than anti-signal, so that we might write the LNS base symbol set as ***{>,<}****.*

Proof. We shall only prove the composition aspect of the LNS base symbol set[a]. Assuming a signal is produced or read. It shall only be expressible as "<", whose only permissible inverse or alternate is ">" — by **Definition 3** and **Definition 2**. No other information encoding symbols exist that can be used in writing a LNS encoded information expression. □

[a] Meaning, we aren't to attempt to prove what the **Ordered Symbol Set** or **Natural Symbol Set**[2] of LNS is at the moment

Note that, to simplify our calculus when dealing with the LNS Base Symbol Set, sometimes we might just write ψ_l instead of ψ_{LNS}.

Theorem 2 (Permissible Information). *Any information when expressed in the LNS shall either be the signal or its alternate — an expression of the absence of signal — the anti-signal.*

Proof. Assuming anything but the signal is expressed, it shall be ">" — for which the only other alternative is the signal (by **Definition 3**), otherwise what is expressed is nothing but the signal, "<" □

Definition 4 (An Information Expression). *An information expression, Θ, is any systematic encoding of meaning as a finite sequence of distinct symbols from a definite symbol set such as ψ_l (refer to* ***Theorem 1****).*

Theorem 3 (All Information Production). *All or any information expressible in the LNS is an information expression producible ex-nihilo or based on prior information (another information expression) via the basic permissible information transformation operations[a]; identity, inversion, duplication, concatenation, reduction and production.*

[a] These shall later be properly defined — refer to **Postulate 1**.

 doi:10.20944/preprints202506.0790.v1

Postulate 1 (LNS Basic Permissible Operations). *By **Theorem** 2, the simplest possible information expressions in LNS are:*

- *Signal:* $<$
- *Anti-Signal:* $>$

*Which, when operated on, can produce new information expressions by **Theorem** 3 thus:*

1. *The Identity Operation,* `id(.)` *— a unary operation:*

   (a) *Signal Identity:*
   $id(<) \implies <$
   (b) *Anti-Signal Identity:*
   $id(>) \implies >$
   (c) *Generally, for any LNS expression* $\Theta = \sigma_1\sigma_2...\sigma_n \quad \forall\sigma_{i\in\mathbb{N}} \in \psi_l$*:*
   $id(\Theta) = id(\sigma_1) \cdot id(\sigma_2)... \cdot id(\sigma_n) = \sigma_1\sigma_2...\sigma_n \implies \Theta$

2. *The Inversion Operation,* `in(.)` *— a unary operation:*

   (a) *Signal Inversion:*
   $in(<) \implies >$
   (b) *Anti-Signal Inversion:*
   $in(>) \implies <$

3. *The Duplication Operation,* `du(.)` *— a unary operation:*

   (a) *Signal Duplication:*
   $du(<) \implies <<$
   (b) *Anti-Signal Duplication:*
   $du(>) \implies >>$
   (c) *In General:* $du(\Theta) = id(\Theta) \cdot id(\Theta) \implies \Theta \cdot \Theta$
   So that, for example: $du(<>) \implies <><>$

4. *The Concatenation Operation,* `co(.)` *— a binary operation:*
   $co(<) \implies co(<, \varnothing) = du(<) \implies <<$
   But also:
   $co(<, <) \implies <<$
   $co(<<, <<) \implies <<<<$
   $co(<, >) \implies <>$

   And Generally:
   $co(\Theta) = du(\Theta) \implies \Theta \cdot \Theta$
   $co(\Theta, \Phi) = id(\Theta) \cdot id(\Phi) \implies \Theta \cdot \Phi$

5. *The Information Reduction Operation,* `re(.)` *— a unary operation:*
   $re(<) \implies <$
   $re(>) \implies >$
   $re(<<) \implies <$
   $re(>>) \implies >$
   $re(<>) \implies <$ *or* $>$

   And Generally: $re(\Theta) \implies \Theta^* \subseteq \Theta$ *where, if* $|\Theta| = n$ *to mean* Θ *contains exactly* $\boldsymbol{n}$ *LNS digits, then* $|\Theta^*| = m$ *and that* $m \leq n \quad \forall m, n \in \mathbb{N}$.

6. *The Information Production Operation,* `pr(.)` *— a unary operation:*
   $pr(\varnothing) \implies <$ *or* $>$ *or* $\varnothing$

 doi:10.20944/preprints202506.0790.v1

It shall be interesting to note that **Postulate 1** raises some interesting problems. For example, away from the simplest LNS expressions such as "<" and ">", shall it be correct to define $in(\cdot)$, the **LNS inversion operation** as:

$$in(\Theta) = in(\sigma^*) \implies in(\sigma_1) \cdot in(\sigma_2) ... \cdot in(\sigma_{n-1}) \cdot in(\sigma_n) \quad \forall \sigma_{i \in \mathbb{N}} \in \{<, >\} \tag{1}$$

Also, in the case of the **LNS reduction operation**, $re(\cdot)$, when or how should we decide which of two or more possible permissible results to produce, generate or return from a call with *multi-symbol* input information expression — such as <><<<><>>> or >><? Essentially, what should be returned in the following computation:

Computation 1. $re(<><<<><>>>) \implies \quad ?$

This problem is actually very important in this LNS theory — it is the kind of problem that brings to mind issues like *how to define autonomous choice* or autonomous decision making or **an autonomous random reduction of some input information?** It might likewise spur other problems like; what is the true nature of entropy? What is the identification of signal fundamentally based on? Also, such a critical problem as; is it the operator that decides or what is operated on, concerning what such a dynamic operation as $re(\cdot)$ ought return?

3. Generation, Perception and Encoding of Information as Lu-Number System Information Expressions

Though focus is mostly on what happens after information naturally or artificially generated or produced becomes available for a system to process (of which, encoding it in some new or alternative way is an example of such basic processing), we shall momentarily look at how such information first becomes available to the processing system - such as the Lu-Number Encoder and Generator in this case.

Also, it shall help to clarify on some important matters concerning some of the LNS operations we have just formalized in **Postulate 1**.

3.0.1. Concerning Information Production/Generation with LNS [2]

First of all, for purposes of not prematurely jumping into or deviating away from our core subject in this present inquiry or formulation, we must appreciate that it is not really the responsibility of a Number Production System or Information Expression Encoding System theory to worry about the first production of information — **creation of information** (or rather, *entropy* — changes in information) *ex-nihilo* for example, or how entropy comes to be in the first place (**how reality expresses itself in some conceptual filter or dimension**) in any system and some *reference frame*, but rather, to provide means to systematically express any such originally produced information (via **a natural entropy source** for example), when it becomes manifest — such as to a **basic signal perceptron** or an IoT sensor **when it registers a change or *infomaton* as raw signal; visual, aural, mechanical, tactile, seismic, electric, chemical, thermal, etc. such as from a natural and physical entropy source or dynamic phenomena.** However, it should further be noted that **such physical entropy sources need not be strictly natural or part of a purely natural phenomena, but could also be virtual — simulated, or rather artificial** — such as for entropy sources implemented in virtual and/or augmented reality systems such as meta-verses,

[2] The emphasis expressed in this section is especially intended to readily bring to mind the important aspects of this current formulation and theory that don't directly fall within the scope of just mathematics or even computer science.

 doi:10.20944/preprints202506.0790.v1

computer games and such. Realistically, there might still be some unanswerable questions within the theoretical, though practically useful LNS for now, but that doesn't necessarily mean it is incomplete[3].

For example, concerning the interesting **information reduction** operation, $re(\cdot)$ in LNS (see **Postulate 1**), we could simplify the operation semantics thus:

$$re(\Theta) \implies \ < \ | \ > \ | \ \Theta \ | \ \Theta^* \subset \Theta \ | \ \varnothing \tag{2}$$

This queer definition of the reduction operation is to allow us to explain how it is possible that where there was more than one possible and/or permissible output from the operation based on the input, we still can express meaningful limits on what the result or *correct* output ought look like and/or mean; **the output is always a subset of the input expression**[4].

And then, concerning the strange **information production** operation, $pr(\cdot)$ in LNS — which, because it essentially is the first way that information ever gets expressed in any LNS production system — where none existed that is — from or via the processing of raw input signal — that might not necessarily be originally expressed using LNS encoding — such as when such information is produced by a natural system, e.g. the thermodynamic and/or kinematic properties of a naked flame/fire, the Brownian motion of gas particles, the kinematics and geometries of a cloud, the hydrodynamics of a fluid under turbulence, certain statistics in a financial market etc., and yet, its output should be information meaningful within the LNS encoding. Thus, we should perhaps generally define $pr(\cdot)$ as such:

$$pr(\cdot) ::= \ < \ | \ > \ | \ \varnothing \tag{3}$$

borrowing notation from the **Backus-Naur form**[5] notation or

$$pr(\cdot) \longrightarrow \ < \ | \ > \ | \ \varnothing \tag{4}$$

borrowing from **CNF grammar/syntax production rules** notation[6] common in the expression of **branching** or **stochastic** production processes in software language engineering literature. Generally, for any number generation founded on the LNS:

$$pr(\cdot) \longrightarrow \ < \ | \ > \ | \ \Theta \ | \ \varnothing \tag{5}$$

for Θ a valid LNS information expression.

[3] The appeal to Gödel's Incompleteness Theorem might seem accidental, however, as we are still just developing this theory for the first time, is perhaps great to keep in mind just in case.

[4] The empty set, $\varnothing$, is likewise implied/expressed, not only because it logically is a valid subset of any input expression, but that, practically or rather, semantically, there might be scenarios in which the function or processing fails to return or halt, and thus no meaningful information can be returned except nothing.

 doi:10.20944/preprints202506.0790.v1

And, before we close discussion concerning **pr(.)**, note that, the choice of notation when expressing such a function is quite important. For example, from a higher abstraction level — say at the level of a user of an RNG function such as the TEA programming language primitive **n:** that is built into the language so as to help generate random numbers[7] — it might be the case that from the user's reference frame or rather, point of view, the basic RNG $pr(\cdot)$ is usable even without any explicit user input — such as for the TEA primitive we have just talked about — meaning, it seems as though the [correct] signature of the **information production operator** ought be written as:

$$pr(\emptyset) \longrightarrow \quad < \quad | \quad > \quad | \quad \Theta \quad | \quad \emptyset \tag{6}$$

So that, it justifies the idea of creating new information from nothing (thus "ex-nihilo") or rather, specifying an RNG that uses/requires no seeds! However, despite the aesthetic gravity of this later signature, it is wiser to stick to the earlier variant (**Equation 5**), or perhaps, we could instead write $pr(\cdot|\emptyset)$ to imply both points of view, though, $pr(\cdot)$ still remains preferable because, apart from helping abstract away what exactly the nature of the arguments the generator expects or operates on, at least it makes it clear there is always some argument no matter what. And in fact, at the lowest level, such as when dealing with a TRNG that say sources entropy from a physical source[8], we find that despite the user of the function not having to explicitly specify an invocation-time argument when seeking for a new [possibly random] information expression, and yet, behind the scenes, at minimum, such an operation must explicitly specify when and from which source (or even perhaps how) the desired information must be read or extracted from the entropy source. Thus, to balance between abstraction, elegance and correct semantics, (**Equation 5**) shall be our most authoritative definition of the generator operation.

4. Generating Binary Numbers from Signal

Now that we have seen how to generate and/or produce simple and complex information expressions such as numbers in the Lu-Number System, we shall next turn our attention to the production of information expressions in a form more palatable and relatable for practical applications in most real-life domains. Essentially, we are to first look at the simplest number system useful in typical computation processes — the base-2 number system, also known as the "binary number system", BNS.

 doi:10.20944/preprints202506.0790.v1

Postulate 2 (Generating Binary Numbers from Signal). *To generate conventional binary numbers from LNS, we simply need to specify the equivalence mapping or correspondence between the BNS and LNS. For all practical purposes, we can safely define such a mapping as such:*

$$< \longleftrightarrow 1 \tag{7}$$

$$> \longleftrightarrow 0 \tag{8}$$

And with just those two mappings defined, the very useful productions expressed in ***Postulate 1*** *still hold for* ***binary information expressions****. For example, we readily shall find that:*

1. *The Identity Operation on Binary Expressions (BE):*

   (a) $id(1) \implies 1$
   (b) $id(0) \implies 0$
   (c) *Generally, for any BE* $\Delta = \delta_1\delta_2...\delta_n \quad \forall\delta_{i\in\mathbb{N}} \in \psi_2$*:*
   $id(\Delta) = id(\delta_1) \cdot id(\delta_2)... \cdot id(\delta_n) = \delta_1\delta_2...\delta_n \implies \Delta$

2. *The Inversion Operation on BE:*

   (a) $in(1) \implies 0$
   (b) $in(0) \implies 1$
   (c) *Generally, for any BE,* $\Delta = \delta_1\delta_2...\delta_n \quad \forall\delta_{i\in\mathbb{N}} \in \psi_2$*:*
   $in(\Delta) = in(\delta_1) \cdot in(\delta_2)... \cdot in(\delta_n)$

   So that it feels as though $in(\cdot)$ *operates on a BE as would a standard* ***bitwise NOT operator***[a]

3. *The Duplication Operation on BEs:*

   (a) $du(1) \implies 11$
   (b) $du(0) \implies 00$
   (c) *In General:* $du(\Delta) = id(\Delta) \cdot id(\Delta) \implies \Delta \cdot \Delta$
   So that, for example: $du(10) \implies 1010$

4. *The Concatenation Operation on BEs:*
   $co(1) \implies co(1, \varnothing) = du(1) \implies 11$
   And like with LNS:
   $co(1, 1) \implies 11$
   $co(11, 11) \implies 1111$
   $co(0, 1) \implies 01$ *— which is* ***not*** *the same as "1", simply because we are dealing with information expressions (strings encoding numbers) and not literal numbers say in their standard or final canonical forms.*

   So, generally, for Δ *and* Λ *that are BEs:*
   $co(\Delta) = du(\Delta) \implies \Delta \cdot \Delta$
   $co(\Delta, \Lambda) = id(\Delta) \cdot id(\Lambda) \implies \Delta \cdot \Lambda$

5. *The Information Reduction Operation on BEs:*
   $re(1) \implies 1$
   $re(0) \implies 0$
   $re(11) \implies 1$
   $re(00) \implies 0$
   $re(10) \implies 1$ *or* 0

   And Generally: $re(\Delta) \implies 1 \mid 0 \mid \Delta \mid \Delta^* \subset \Delta \mid \varnothing$

6. *The Information Production Operation for BEs:*
   $pr(\cdot|\varnothing) \longrightarrow 1 \mid 0 \mid \Delta \mid \varnothing$

[a] Such as the operator $\sim$ in Python, which, when invoked as such: $\sim$ 101 should return "010" — e.g with `a=0b101;b= a&0b111;print(bin(b))`

 doi:10.20944/preprints202506.0790.v1

Lemma 1 (Equivalence of ψ_1 and ψ_2). *By **Postulate** 2 and the fact that the Base-2 natural symbol set[2] is merely $\langle 0, 1 \rangle$, if we write the LNS base symbol set (see **Theorem** 1) in its ordered form as $\psi_1 = \langle >, < \rangle$, then $\psi_1 \equiv \psi_2$.*

The interesting and useful consequences of **Lemma 1** is that any typical binary number expressions (or rather, information expressions encoded using bits) can all be re-written simply as expressions in the Lu-Number System. Or rather, at the symbol-set level then, the two seemingly different information encoding number systems are equivalent and expressions in one system can readily be re-written in the other. Illustrations of this would be transforms such as:

1. $101 \longleftrightarrow <><$
2. $000111 \longleftrightarrow >>><<<$
3. $<><<<> \longleftrightarrow 101110$
4. $><>< \longleftrightarrow 0101$

5. Generating Decimals from Signal

Now that we have seen the case of obtaining binary numbers from signal, or rather, via the Lu-Number System, and having seen the interesting property that the BNS and LNS are equivalent in some regards, we can then turn our attention to information expressions of numbers in base-10 and see what interesting things we might find out based on LNS.

Postulate 3 (Decimals from Signal). *To produce conventional base-10 numbers starting from the LNS, we basically process raw signal from an entropy source (physical or not, natural or artificial does not matter), into an information expression encoded in LNS as usual (**Postulate** 1). This then, can be remapped into binary (see **Postulate** 2), and using standard base-conversion mathematics — either bitwise or while treating of the entire BE as a canonical form number in binary (see **Definition** 1), or via chunking — the BE thus produced can be transformed into a base-10 information expression.*

Next, we shall look at the generation of numbers in any base.

6. A General Number Generator and the True Random Number Generator

Building on the ideas we have seen in the previous sections, we can safely trust that the LNS can readily allow us to design a number generator targeting any base starting from the processing of some basic signal.

Postulate 4 (Generating Numbers in Any Base). *The process laid out in **Postulate** 3 can be used to generate number expressions for bases other than 10, simply by building upon the decimal information expression produced using that method, and then using conventional base-conversion maths to convert from decimal to any desired target base.*

Turning our attention away from the generation of just **any** information expression, or rather, any numeric expressions or numbers, we shall now look at the more important matter of how to generate **random** information expressions and consequently, how to generate **truly random numbers**.

 doi:10.20944/preprints202506.0790.v1

Definition 5 (**A True Random Number Generator (TRNG)**). *A true random number generator is any systematic process capable of operating on the output of an entropy [or chaos] source — **some information source that might or might not directly produce anything but simple information expressions such as basic signal or its absence**[a], **in a random/equiprobable and statistically independent manner, and which source is [preferably totally] independent of the information processing system**, and which information thus produced and then read/perceived, becomes encoded as the input information for the generation/production process — in which case such information might as well be considered "the seed", and which is then used to systematically, algebraically or computationally produce more meaningful or rather more complex information in the form of an information expression in some particular/more desirable number/encoding system.*

[a] This has been well introduced in **Section 3.0.1**

At this juncture, we can pause and appreciate that any random number generator is in one way or another, an information processor — more correctly, a signal encoder. However, for TRNGs, the conditions laid out in **Definition 5** are undeniably critical and important, since, they shall help us to readily tell apart RNGs that are truly/appreciably random from those that are only-approximately random (such as *pseudo-random number generators*, PRNGs), that might forego use of an external, statistically independent and semantically correct source of randomness, and instead rely on user-provided seeds, say for purposes of allowing the generation of predictable random numbers or sequences of them, and thus, produce or generate predictable information/results — see [8] for more on this). This shall also help in the design and implementation of special RNGs meant for critical operations such as in enabling cryptography or information security via the use of Cryptographically Secure Random Number Generators (CSRNGs).

Theorem 4 (**All TRNGs Express Random Signal**). *All semantically correct TRNGs shall obey or operate on some independent and external random signal so as to produce a number or information expression that is adequately random.*

Proof. As long as the source is of a stochastic kind[a] — which is beyond the scope of this proof and/or present work, for TRNGs based on physical entropy sources, this is guaranteed by **Postulate 1**, otherwise by **Postulate 4**. □

[a] It shall be interesting to note that in practice, even where the entropy source is not necessarily of a high-quality kind or rather not one obeying a uniform distribution — one of the fundamental properties of an appreciably and/or statistically sound randomness source, it still might be the basis of a TRNG if the signal thus obtained from it is used as input into a more sophisticated process that can operate on it further, so as to turn the otherwise non-uniform entropy signal into one that is — see Bikos Anastasios et al [8] for more on this. But also, there are practical scenarios in which the generation of just any random number isn't want is desired, but rather, the generation of random numbers conforming to some particular distribution such as Gaussian/Normal for example — refer to DB Thomas et al.[9]

Next, we shall attempt to distinguish between RNGs that operate directly on raw signal (physical information) vs those that operate on complex signal — or rather, on finite strings such as [higher-order] information expressions (see **Definition 4**).

Definition 6 (**First Order RNG, Ξ**). *A first order RNG, Ξ, is any function*

$$\Xi(\cdot|\varnothing) = \sum_{i=0}^{\infty} \mu$$

*Where $\mu \in \psi_l$ and the **addition operation**, $+$, is redefined such that $\mu + \mu = co(\mu)$ and generally, $\forall \sigma, \mu \in \psi_l, \quad \sigma + \mu = co(\sigma, \mu) = \sigma\mu$ as laid out in **Postulate 1**.*

We shall then find that the production operator in LNS, $pr(\cdot)$ is actually a kind of Ξ.

 doi:10.20944/preprints202506.0790.v1

Definition 7 (Higher Order RNG, ξ). *A higher order RNG, ξ, is any function*

$$\xi(\cdot|\Theta|\varnothing) = \sum_{i=0}^{n} \sigma\beta^i$$

*Where $\sigma \in \psi_\beta$ and the **addition operation**, $+$, is redefined such that $\sigma\beta^i + \sigma\beta^i = co(\sigma\beta^i)$ and generally, $\forall \sigma \in \psi_\beta$ and $\mu \in \psi_\alpha$, $\sigma\beta^i + \sigma\alpha^j = co(\sigma\beta^i, \sigma\alpha^j)$, where $\alpha, \beta, i, j \in \mathbb{N}$ and Θ is some information expression.*

Lemma 2 (All ξ-TRNG Reducible to Ξ-TRNG). *Any higher-order TRNG producing numbers in any base or number system can be reduced to a simple first-order TRNG in the LNS.*

7. Conclusion

In this work, we have developed a new number system, LNS, a basic, first-order information encoding system fit for expressing basic outputs from an entropy source — especially physical entropy sources, in the form of signal or its absence. LNS provides a low-level framework upon which interaction with entropy sources can be made so that higher-level information expressions such as numbers in binary, decimal and other usual bases can be readily obtained or inferred from mere basic signal. We have seen how the basic LNS number generator would work from a purely theoretical perspective, and have likewise seen how other higher order number generators — in particular, Binary Number Generators and then Decimal Number Generators can be founded on an LNS generator. Apart from offering the foundations for rigorously specifying a basic RNG, via specifying the permissible basic operations possible on LNS-encoded basic information, this work also contributes to the literature on especially TRNGs, by establishing how any TRNG irrespective of its target or output number system, can be founded on, or rather, can be reduced to a basic LNS-TRNG — a $\Xi(\cdot|\varnothing)$. In future work, we shall further this undertaking with pragmatic verification of the theory thus laid out, say with a reference implementation of a LNS-TRNG, and leveraging that to implement a high-order RNG, $\xi(\cdot|\Theta|\varnothing)$, such as a decimal TRNG, so as to practically verify this theory and also help extend it further. Also, more theoretical work might delve into answering some of the still-unanswered questions raised in this present work, such as how exactly to specify when/how a typical branching production rule in a generator specification might rely on some explicit condition so as to decide how to produce one of several possible output [terminal or non-terminal] terms, a solution that might likewise be of theoretical and practical relevance in automated language engineering such as in implementing systems than can generate valid source-code/computer programs autonomously or even stochastically, based on processing of a language specification or formal grammar (BNF or CNF production rules for example), and which are driven by some stochastic signal or input — ideas useful in cases such as if we are to work on automated-generation of valid and useful TEA programs ex-nihilo.

References

1. Joseph Willrich Lutalo. A general theory of number cardinality. *Academia.edu*, Jan 2024. *Accessible via* https://www.academia.edu/43197243/A_General_Theory_of_Number_Cardinality.
2. Joseph Willrich Lutalo. Concerning a special summation that preserves the base-10 orthogonal symbol set identity in both addends and the sum. 2025. *Accessible via* https://www.academia.edu/download/122499576/The_Symbol_Set_Identity_paper_Joseph_Willrich_Lutalo_25APR2025.pdf.
3. Joseph Willrich Lutalo. Numbers from arbitrary text: Mapping human readable text to numbers in base-36. *Academia.edu*, 2024. *Accessible via* https://www.academia.edu/123296302/Numbers_from_Arbitrary_Text_Mapping_Human_Readable_Text_to_Numbers_in_Base_36.
4. Michael J. Bossé and William J. Cook. Repeating decimal expansions in different bases. *Electronic Journal of Mathematics & Technology*, 2025. *Accessible via* https://ejmt.mathandtech.org/Contents/eJMT_v16n3p4.pdf.
5. Donald E. Knuth. Backus normal form vs. backus-naur form. *Communications of the ACM*, 7(12):735–736, 1964. *Accessible via* https://dl.acm.org/doi/pdf/10.1145/355588.365140.

 doi:10.20944/preprints202506.0790.v1

6. V Aho Alfred, S Lam Monica, and D Ullman Jeffrey. *Compilers principles, techniques & tools*. pearson Education, 2007. *Accessible via* https://elib.vku.udn.vn/bitstream/123456789/2542/1/2007.%20Compilers-Principles%2C%20Techniques%2C%20and%20Tools%20%282nd%20Edition%29.pdf.
7. Joseph Willrich Lutalo. *TEA TAZ - Transforming Executable Alphabet A: to Z: COMMAND SPACE SPECIFICATION*, 2024. Accessed on 14 May, 2025, via https://www.academia.edu/122871672/TEA_TAZ_Transforming_Executable_Alphabet_A_to_Z_COMMAND_SPACE_SPECIFICATION.
8. Anastasios Bikos, Panagiotis E Nastou, Georgios Petroudis, and Yannis C Stamatiou. Random number generators: Principles and applications. *Cryptography*, 7(4):54, 2023. *Accessible via* https://www.mdpi.com/2410-387X/7/4/54.
9. David B Thomas, Wayne Luk, Philip HW Leong, and John D Villasenor. Gaussian random number generators. *ACM Computing Surveys (CSUR)*, 39(4):11–es, 2007. *Accessible via* http://cas.ee.ic.ac.uk/people/dt10/research/thomas-07-gaussian-survey.pdf.