ISBN: 978-628-96613-3-0. ISSN: 2414-6390. Digital Object Identifier: https://dx.doi.org/10.18687/LACCEI2026.1.1.502

Santiago Length Dynamical System to Predict Message Inflation and Size Buffers in IIoT Pipelines

Mendoza Delgado, Raphael Santiago, Estudiante[1]; Mendoza Arenas, Ruben Dario, Doctor[1]; Rojas Orbegoso, Jorge Luis, Magister[1]; Farfán Aguilar, José Antonio, Doctor[1]; Morales Chalco, Osmart Raúl, Doctor[1]; La Chira Loli, Monica Beatriz, Doctora[2]; Arbañil Rivadeneira, Rubén Orlando, Doctor[3]; [1]Universidad Nacional del Callao, Perú, rsmendozad@unac.edu.pe, rdmendozaa@unac.edu.pe, jlrojaso@unac.edu.pe, jafarfana@unac.edu.pe, ormoralesc@unac.edu.pe, [2]Universidad Autónoma del Perú, Perú, monica.lachira@autonoma.pe, [3]Universidad Nacional Mayor de San Marcos, Perú, rarbanilr@unmsm.edu.pe.

Abstract- We propose the Santiago Dynamical System (SDS) as a unifying formulation of discrete dynamics on $\mathbb{N}$ *generated by a coding and an observable. The central case studied in this work is the* length SDS*, where the observable is the length of a word* $\alpha(n)$ *over an alphabet* Σ*, giving rise to the iteration* $n_{k+1} = |\alpha(n_k)|$*. This perspective connects functional graphs on* $\mathbb{N}$ *with symbolic dynamics, coding theory, and Lyapunov-type potential methods. Universal structural properties are presented, including contraction/expansion criteria and a methodological framework to characterize attractors, basins, and iterative complexity in classes of natural codings. Finally, conjectures are formulated concerning small attractors and iterated-logarithm-type convergence in compact codings.*

Keywords: discrete dynamical systems, symbolic dynamics, coding theory, word length, functional graphs, Lyapunov methods.

Sistema Dinámico de Santiago de Longitud para Predecir Inflación de Mensajes y Dimensionar Buffers en Pipelines IIoT

Resumen—Se propone el Sistema Dinámico de Santiago (SDS) como una formulación unificadora de dinámicas discretas en $\mathbb{N}$ generadas por una codificación y un observable. El caso central de este trabajo es el SDS de longitud, donde la observable es la longitud de una palabra $\alpha(n)$ sobre un alfabeto Σ, dando lugar a la iteración $n_{k+1} = |\alpha(n_k)|$. Esta perspectiva conecta grafos funcionales en $\mathbb{N}$ con dinámica simbólica, teoría de códigos y métodos de potencial tipo Lyapunov. Se presentan propiedades estructurales universales, criterios de contracción/expansión y un esquema metodológico para caracterizar atractores, cuencas y complejidad iterativa en clases de codificaciones naturales. Finalmente, se formulan conjeturas sobre atractores pequeños y convergencia tipo logaritmo iterado en codificaciones compactas.

Palabras clave: sistemas dinámicos discretos, dinámica simbólica, teoría de códigos, longitud de palabra, grafos funcionales, métodos de Lyapunov.

I. Introducción

El estudio de sistemas dinámicos discretos de la forma (X, f) con $f : X \to X$ constituye un eje central tanto en matemática pura como aplicada, al capturar procesos iterativos y fenómenos de estabilidad, periodicidad y complejidad [13], [11], [12]. En contextos aritméticos, el caso $X = \mathbb{N}$ y $f : \mathbb{N} \to \mathbb{N}$ produce un objeto combinatorio natural: el *grafo funcional* (cada vértice tiene exactamente una arista saliente), cuya geometría codifica puntos fijos, ciclos y cuencas de atracción [16]. Esta representación es particularmente eficaz para estudiar *periodicidad eventual*: cuando la órbita permanece en un conjunto finito, necesariamente repite estados y cae en un ciclo (principio del palomar) [12].

Sin embargo, muchas dinámicas en $\mathbb{N}$ provienen de procedimientos que primero *transforman* el entero en una representación simbólica y luego aplican una lectura numérica: por ejemplo, la longitud de la escritura en base b, el peso de una representación en un código, la complejidad de un mensaje o la longitud de un nombre en lenguaje natural. En tales casos, el estudio directo de f oculta la estructura que emerge de la codificación. Esta observación motiva el SDS: en lugar de postular $S : \mathbb{N} \to \mathbb{N}$ arbitrariamente, se lo modela como una composición de dos pasos,

$$\mathbb{N} \xrightarrow{\alpha} \mathscr{W} \xrightarrow{\Phi} \mathbb{N}, \qquad S = \Phi \circ \alpha, \tag{1}$$

donde α es una codificación y Φ una observable. Esta factorización es reminiscentemente cercana a (i) la dinámica simbólica, donde se representa un sistema por secuencias o palabras [14], [15], (ii) la teoría de la información, donde se mide longitud de descripciones y códigos [2], [1], y (iii) perspectivas de complejidad algorítmica, donde la longitud de una descripción es un recurso [4].

En este artículo, se privilegia el caso en el que la observable es la *longitud* de la codificación: si Σ es un alfabeto y Σ^* el conjunto de palabras finitas, entonces $\mathscr{W} \subseteq \Sigma^*$ y

$$\Phi(w) = |w|, \qquad S(n) = |\alpha(n)|. \tag{2}$$

La iteración asociada adopta la forma

$$n_{k+1} = S(n_k) = |\alpha(n_k)|, \quad k \geq 0. \tag{3}$$

Este mecanismo es conceptualmente simple, pero dinámicamente amplio: de hecho, cualquier función $F : \mathbb{N} \to \mathbb{N}_0$ puede representarse como un SDS de longitud tomando una codificación unaria $\alpha(n) = a^{F(n)}$ [4]. Esta *universalidad* implica que la teoría total del SDS sin restricciones equivale a la teoría de todas las autoaplicaciones en $\mathbb{N}$; por ello, el punto matemático decisivo consiste en fijar *clases naturales* de codificaciones (numerales, lingüísticas, prefijo-libre, computables, etc.) donde aparezcan fenómenos robustos y teoremas no triviales [2], [3].

La longitud como observable es relevante por varias razones. Primero, es una medida canónica de tamaño: en teoría de códigos, minimiza costos de transmisión y almacenamiento [2], [3]; en dinámica simbólica, el crecimiento de longitud se relaciona con complejidad combinatoria [8], [9]; y en informática teórica, la longitud se conecta con recursos computacionales y con la complejidad descriptiva [5], [4]. Segundo, la iteración de una transformación "longitud de la codificación" tiende a producir *contracción* en codificaciones compactas: por ejemplo, la longitud de la escritura posicional es $\Theta(\log n)$, lo que sugiere caídas rápidas hacia un conjunto pequeño de estados y, por tanto, periodicidad eventual [10], [12]. Tercero, esta formulación habilita un arsenal de métodos de prueba tipo Lyapunov: si existe un potencial V que decrece bajo S, se obtiene atrapamiento en un conjunto finito y conclusiones dinámicas fuertes [13], [12].

Existe, además, una analogía conceptual con problemas clásicos de iteración aritmética que exhiben fenómenos de "contracción irregular" y atractores pequeños; un caso emblemático es la dinámica de Collatz, donde la complejidad surge de una regla simple [17]. Si bien el SDS de longitud es distinto, comparte el enfoque de estudiar *clases* de aplicaciones sobre enteros con preguntas centrales: *(i) toda órbita termina en un ciclo?, (ii) cuáles son los atractores?, (iii) cuán rápido se llega a ellos?, (iv) cuán sensibles son los resultados a la definición de la codificación?* Tales preguntas aparecen también en dinámicas definidas por representaciones digitales (por ejemplo, mapas de suma de dígitos) y en procesos de normalización o compresión [10], [3].

En particular, para codificaciones lingüísticas (p. ej., nombres de números en español), la longitud depende de reglas ortográficas y morfológicas; esto abre un puente entre dinámica discreta y gramáticas formales [7], [6], e introduce un aspecto metodológico importante: *la longitud es sensible a*

la convención (contar o no espacios, tildes, guiones, etc.). En este trabajo, en lugar de fijar una convención universal, explicitamos el *protocolo de longitud* como parte del modelo y analizamos su impacto, siguiendo el espíritu de reproducibilidad en modelos basados en reglas [5].

El aporte principal de este artículo no consiste en proponer un nuevo algoritmo de compresión ni una arquitectura industrial completa para IIoT, sino en formular un marco matemático-computacional para modelar la evolución del tamaño de mensajes mediante sistemas dinámicos discretos de longitud. En particular, el SDS permite representar procesos repetidos de codificación, encapsulación, compresión, normalización o cifrado mediante una dinámica de la forma $n_{k+1} = S(n_k)$.

Desde esta perspectiva, las aplicaciones en pipelines IIoT se presentan como escenarios de validación computacional del modelo propuesto. Por tanto, el alcance del trabajo se centra en analizar estabilización, inflación de mensajes, aparición de atractores y criterios preliminares para dimensionar buffers. La implementación en infraestructuras industriales reales y la calibración con tráfico capturado en campo quedan delimitadas como líneas futuras de investigación.

I-A. Contribuciones

Las contribuciones se organizan en cuatro líneas:

- Se formaliza el SDS de longitud y su relación con grafos funcionales y dinámica simbólica, destacando criterios universales de periodicidad eventual [12], [14].
- Se establecen criterios de contracción/expansión y un método de potencial que garantizan la existencia de atractores finitos y controlan convergencia [13].
- Se propone una metodología teórico-computacional para identificar atractores, cuencas y tiempos de llegada, usando detección de ciclos y muestreo en clases de codificaciones [16].
- Se formulan conjeturas (atractores pequeños, convergencia tipo $\log^*$ y robustez frente a convenciones) orientadas a un programa de investigación del SDS en codificaciones naturales [2], [4].

I-B. Organización

La Sección II define el marco matemático y el diseño metodológico. La Sección III presenta resultados teóricos y ejemplos prototípicos (codificación posicional, códigos compactos, codificación lingüística). La Sección IV discute implicancias, limitaciones y comparaciones con literatura clásica. La Sección V resume conclusiones y recomendaciones.

II. Materiales y Métodos

II-A. Estructura formal del SDS de longitud

Sea Σ un alfabeto y Σ^* el conjunto de palabras finitas. La longitud $|\cdot| : \Sigma^* \to \mathbb{N}_0$ se define por $|\varepsilon| = 0$ y $|uv| = |u| + |v|$ [8]. Una codificación es una función total

$$\alpha : \mathbb{N} \to \Sigma^*.$$

El SDS de longitud es la aplicación $S : \mathbb{N} \to \mathbb{N}_0$ dada por (2). La órbita desde n_0 queda determinada por (3). Para vincular el lado simbólico, definimos $w_k := \alpha(n_k)$; entonces $n_{k+1} = |w_k|$ y la dinámica induce una transformación en palabras:

$$\beta(w) := \alpha(|w|), \qquad w \in \Sigma^* \tag{4}$$

con lo cual $w_{k+1} = \beta(w_k)$ y $|w_{k+1}| = S(|w_k|)$ cuando se extiende el dominio apropiadamente [14], [15].

II-B. Clases de codificaciones estudiadas

Como el SDS es universal, se requiere restringir α a clases *naturales*. Consideramos tres familias:

F1) Codificación posicional (digital). $\alpha_b(n)$ es la representación en base $b \geq 2$ sin ceros a la izquierda. Entonces $S_b(n) = |\alpha_b(n)| = \lfloor \log_b n \rfloor + 1$ [10].

F2) Codificación de códigos compactos. α es un código prefijo-libre o casi óptimo para una distribución dada (Huffman, Shannon–Fano) y se estudia el crecimiento de longitudes en promedio o peor caso [2], [3].

F3) Codificación lingüística. $\alpha_{\text{es}}(n)$ es un nombre textual del número bajo una gramática fija. La longitud depende del protocolo (espacios, tildes, etc.), y se formaliza como una función de conteo sobre una cadena generada por reglas [7], [6].

II-C. Métricas y observables dinámicas

Para una órbita (n_k) se cuantifican:

(M1) Tiempo de entrada a umbral M:

$$\tau_M(n_0) = \text{ínf}\{k \geq 0 : \ n_k \leq M\}.$$

(M2) Exponente logarítmico superior:

$$\lambda^+(n_0) = \limsup_{k\to\infty} \frac{1}{k} \log(1 + n_k),$$

útil para distinguir crecimiento subexponencial vs exponencial [12].

(M3) Atractor y período: el par (r, p) con $n_{r+p} = n_r$ y p mínimo (si existe) determina un atractor cíclico [16].

II-D. Detección de ciclos y protocolo computacional

Para detectar periodicidad sin almacenar toda la historia, se usa el método de Floyd (tortuga-liebre), estándar en iteraciones [10]. El Algoritmo 1 entrega r y p.

II-E. Diseño de experimentos

En la parte computacional se propone:

- *Barrido* de n_0 en intervalos crecientes para estimar cuencas por frecuencias.
- *Exploración de atractores* en conjuntos atrapantes B (cuando hay contracción), construyendo el subgrafo inducido por S sobre B [16].
- *Análisis de sensibilidad* ante cambios en el protocolo de longitud (p. ej. contar espacios o no) para codificación lingüística [6].

Algorithm 1 Detección de ciclo (Floyd) para $n_{k+1} = S(n_k)$.

Input: S, n_0
$x \leftarrow S(n_0)$; $y \leftarrow S(S(n_0))$
while $x \neq y$ **do**
  $x \leftarrow S(x)$
  $y \leftarrow S(S(y))$
end while
/* Encontrar inicio del ciclo */
$\mu \leftarrow 0$; $x \leftarrow n_0$
while $x \neq y$ **do**
  $x \leftarrow S(x)$; $y \leftarrow S(y)$
  $\mu \leftarrow \mu + 1$
end while
/* Encontrar longitud del ciclo */
$\lambda \leftarrow 1$; $y \leftarrow S(x)$
while $x \neq y$ **do**
  $y \leftarrow S(y)$
  $\lambda \leftarrow \lambda + 1$
end while
Output: $r = \mu,\ p = \lambda$

II-F. Cómo modelar procesos reales como un SDS *de longitud*

En aplicaciones, el estado $n \in \mathbb{N}$ se interpreta como una magnitud discreta de 'tamaño" (por ejemplo, número de símbolos, bytes, tokens, caracteres o longitud de un identificador) y la codificación α representa un *proceso de transformación/representación* que produce una palabra $w = \alpha(n)$ en un alfabeto Σ (bits, bytes, caracteres, tokens, etc.). La longitud $|w|$ se convierte en el nuevo estado, dando $n^+ = S(n) = |\alpha(n)|$. Este esquema captura escenarios de:

- **Encadenamiento de capas de representación:** encapsulación de protocolos, serialización, codificación de enteros, o empaquetado de mensajes [34].
- **Transformaciones de compresión/normalización:** compresión, tokenización, normalización de texto o canonización de identificadores [31], [32].
- **Pipelines de ingeniería de datos:** ETL/ELT donde los objetos pasan por formatos sucesivos (JSON→binario→compresión), generando una dinámica de longitudes [33].

La iteración del SDS corresponde a repetir el procedimiento (por diseño o por error operativo): volver a comprimir, volver a encapsular, re-serializar, re-tokenizar, o re-normalizar. Esta repetición es común en práctica (p. ej. 'comprimir lo ya comprimido") y suele conducir a atractores pequeños (longitudes estables) o a crecimientos por sobrecarga (headers, metadatos) [31], [32].

II-G. Evaluación computacional en escenarios IIoT

Para fortalecer la validación práctica del SDS de longitud, se considera un escenario computacional aplicado a pipelines IIoT, donde el estado n_k representa el tamaño del mensaje en bytes antes de la k-ésima transformación, y $n_{k+1} = S(n_k)$ representa el tamaño resultante luego de aplicar serialización, encapsulación, compresión, cifrado y alineamiento.

Se utiliza el modelo operativo:

$$S(n) = B\left\lceil \frac{c(n+h)+t}{B} \right\rceil,$$

donde h representa el overhead fijo por cabeceras y metadatos, c el factor efectivo de compresión, t los bytes adicionales de autenticación o verificación, y B el tamaño de bloque asociado al alineamiento o padding.

La evaluación considera tres escenarios: compresión efectiva, dato casi incompresible y re-encapsulación de mensajes ya procesados. En cada caso se analizan tamaños iniciales representativos y se observan el tamaño máximo alcanzado, la estabilización o inflación del mensaje y la utilidad del modelo para estimar requerimientos preliminares de buffer.

III. Resultados

III-A. Resultados estructurales universales

Proposición 1 (Acotación ⇒ periodicidad eventual). Si la órbita $\mathcal{O}^+(n_0) = \{S^k(n_0) : k \geq 0\}$ es finita, entonces existe $r \geq 0$ y $p \geq 1$ tales que $S^{r+p}(n_0) = S^r(n_0)$.

Demostración (moderada). Si la órbita es finita, existe M con $S^k(n_0) \in \{0,1,\ldots,M\}$ para todo k. En una sucesión infinita con valores en un conjunto finito, existen $0 \leq i < j$ con $S^i(n_0) = S^j(n_0)$. Sea $r = i$ y $p = j - i$. Entonces $S^{r+p}(n_0) = S^r(n_0)$ y, por iteración, $S^{r+mp}(n_0) = S^r(n_0)$ para todo $m \geq 1$, lo que define periodicidad eventual [12]. □

Teorema 1 (Criterio de contracción). Si existe M tal que para todo $n > M$ se cumple $S(n) < n$, entonces toda órbita entra en $\{0,1,\ldots,M\}$ y es eventualmente periódica.

Demostración. Si $n_k > M$, entonces $n_{k+1} = S(n_k) < n_k$, por lo que (n_k) decrece estrictamente mientras se mantenga por encima de M. En particular, debe entrar en $\{0,\ldots,M\}$ en tiempo finito. Luego aplica Proposición 1 [12], [13]. □

Teorema 2 (Criterio de expansión). Si existe N tal que para todo $n \geq N$ se cumple $S(n) \geq n+1$, entonces toda órbita que alcance un valor $\geq N$ diverge (crece sin cota).

Demostración. Una vez que $n_k \geq N$, se tiene $n_{k+1} \geq n_k + 1$, así que la sucesión es estrictamente creciente y no acotada [11]. □

III-B. Cota combinatoria de longitud para codificaciones inyectivas

Un principio de teoría de códigos establece que sobre un alfabeto finito la cantidad de palabras de longitud acotada crece exponencialmente con la longitud [2], [8].

Proposición 2 (Cota inferior de longitud). Sea $|\Sigma| = q \geq 2$ y $\alpha : \mathbb{N} \to \Sigma^*$ inyectiva. Entonces, para todo $n \in \mathbb{N}$,

$$|\alpha(n)| \ \geq\ \left\lceil \log_q\big((q-1)n+1\big)\right\rceil - 1. \tag{5}$$

Demostración (moderada). El número de palabras de longitud $\leq L$ es $1+q+q^2+\cdots+q^L = \frac{q^{L+1}-1}{q-1}$ [8]. Si α es inyectiva, los valores $\alpha(1),\ldots,\alpha(n)$ son n palabras distintas, por lo que necesariamente $n \leq \frac{q^{L+1}-1}{q-1}$ donde $L = \max_{1\leq i\leq n}|\alpha(i)|$.

Cuadro I
ESTRUCTURA CUALITATIVA DEL SDS DIGITAL $S_b(n) = \lfloor \log_b n \rfloor + 1$.

Rango	Consecuencia dinámica
$n \geq b^2$	Contracción estricta: $S_b(n) < n$ (entrada a región pequeña).
$1 \leq n < b$	$S_b(n) = 1$ (fuerte colapso a 1).
Región pequeña	Atractores dentro de $\{1, 2, \ldots, \lfloor \log_b(b^2) \rfloor + 1\}$; típicamente fijos.

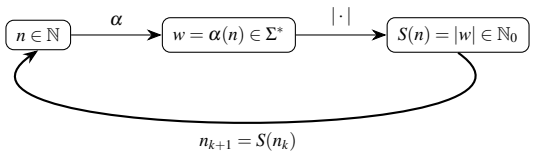

Figura 1. Esquema conceptual del SDS de longitud.

Reordenando se obtiene $q^{L+1} \geq (q-1)n+1$, lo que implica $L \geq \log_q((q-1)n+1) - 1$, y al tomar techo se llega a (5). □

III-C. Ejemplo prototípico: longitud digital en base b

Para α_b representación en base b, se obtiene

$$S_b(n) = \lfloor \log_b n \rfloor + 1.$$

Este mapa es fuertemente contractivo: para $n \geq b^2$, se cumple $S_b(n) \leq \log_b(n) + 1 < n$ [10]. Por Teorema 1, toda órbita entra rápidamente a un conjunto pequeño y por tanto termina en un atractor dentro de ese conjunto. En la práctica, los atractores suelen ser puntos fijos o ciclos muy cortos, dependiendo de la convención para n pequeño.

III-D. SDS de longitud con códigos compactos

En códigos de compresión, las longitudes se relacionan con probabilidades y entropía [1], [2], [3]. Si α es un esquema de codificación que satisface una cota superior de la forma

$$|\alpha(n)| \leq A \log(n+1) + B \quad (n \geq n_0), \tag{6}$$

entonces, para n grande, $S(n) < n$, de donde se deduce periodicidad eventual por Teorema 1. Más aún, la cota (6) sugiere tiempos de caída extremadamente pequeños, del orden de $\log^*$, el número de iteraciones del logaritmo necesarias para caer bajo un umbral fijo [4], [5].

III-E. SDS lingüístico: protocolo de longitud y sensibilidad

Sea $\alpha_{\text{es}}(n)$ una palabra en español generada por una gramática fija, y sea $|\cdot|_\pi$ la longitud bajo un protocolo π (por ejemplo: contar letras sin espacios, o incluir espacios, o normalizar tildes). Entonces el SDS es

$$S_{\text{es},\pi}(n) = |\alpha_{\text{es}}(n)|_\pi.$$

La teoría formal indica que cambios en π pueden alterar atractores y cuencas, incluso si la gramática subyacente es la misma [6], [7]. Esto motiva reportar siempre π de forma explícita como parte del modelo.

III-F. Aplicaciones a la vida real del SDS de longitud

El SDS de longitud funciona como un *modelo de caja negra* para estudiar cómo evoluciona el tamaño de representaciones cuando se aplican repetidamente mecanismos de codificación. En esta subsección se describen aplicaciones concretas donde el estado n_k es una longitud relevante (caracteres, bytes o tokens) y la iteración captura ciclos reales de procesamiento [2], [3], [31].

III-F1. Compresión, almacenamiento y "compresión de lo comprimido": En compresión sin pérdida, un archivo se transforma en una secuencia de bits con metadatos (cabeceras, tablas, checksums). En muchos sistemas, por rutinas automáticas o por uso incorrecto, un archivo ya comprimido vuelve a comprimirse (p. ej. aplicar un compresor sobre un contenedor comprimido). En términos del SDS, se modela con:

$$S(n) = |\alpha(n)|. \tag{7}$$

donde $\alpha(n)$ denota el resultado de aplicar el compresor a un objeto de tamaño n.

La teoría predice dos comportamientos observables:

- **Atracción a longitud estable:** si el algoritmo se acerca a una forma "incompresible", la longitud no baja significativamente y puede estabilizarse (punto fijo aproximado) [31], [32].
- **Aumento por sobrecarga:** en datos ya comprimidos, la nueva capa puede añadir metadatos, causando $S(n) \geq n$ para rangos grandes (tendencia expansiva local) [31].

Este análisis es útil para diseñar políticas operativas: detectar pipelines que recomprimen innecesariamente, estimar costos de almacenamiento y prever el *punto de saturación* donde volver a transformar deja de ser beneficioso [33].

III-F2. Redes, encapsulación y sobrecarga de protocolos: En redes, un mensaje atraviesa capas (aplicación, transporte, red, enlace) que agregan cabeceras y, a veces, padding. Si n es el tamaño del payload, la codificación $\alpha(n)$ representa el paquete final serializado. Entonces $S(n)$ modela el tamaño *en el cable*. Iterar S representa encapsulación repetida (túneles, VPNs, capas adicionales, re-empaquetado) y permite estudiar:

- umbrales donde el overhead domina,
- condiciones de expansión ($S(n) \geq n + c$) que implican crecimiento inevitable,
- existencia de regiones "estables" cuando el sistema ajusta MTU/segmentación y la longitud efectiva se estabiliza por reglas de fragmentación [34].

III-F3. Ingeniería de datos: ETL/ELT, logs y serialización: En sistemas de datos, un registro puede pasar por múltiples transformaciones: limpieza, normalización, serialización (p. ej. JSON), compresión y empaquetado. El SDS modela el efecto global de repetir un pipeline por re-procesamientos o reintentos automáticos:

$$n_{k+1} = S(n_k) = |\alpha(n_k)|.$$

Con esto, se pueden estimar:

- costos de almacenamiento al re-procesar datos,
- riesgo de inflación de logs (si hay metadatos repetidos),

- diseño de reglas para evitar "crecimiento por re-etiquetado" [33], [32].

III-F4. Ciberseguridad: representaciones de longitud fija y estabilización: En criptografía, muchas primitivas producen salidas de longitud fija (hashes, MACs, ciertos cifrados por bloques en modos adecuados). Si se modela $\alpha(n)$ como el resultado de aplicar una transformación criptográfica a un mensaje de longitud n, entonces $S(n)$ suele ser constante o casi constante (dependiendo del esquema), lo cual genera un atractor fuerte (punto fijo de longitud) [35]. Esto es útil para:

- auditar pipelines: detectar pasos donde la longitud deja de reflejar tamaño real (pérdida de información bajo la proyección longitud),
- diseñar sistemas de almacenamiento de huellas digitales con costos previsibles [35].

III-F5. Educación y NLP: longitud de nombres, tokenización y complejidad textual: En educación y procesamiento de lenguaje natural, $\alpha(n)$ puede ser el *nombre en lenguaje natural* de n o una tokenización de un texto de tamaño n. El SDS permite estudiar iteraciones de "re-expresión" (parafraseo, normalización, re-tokenización) midiendo cómo evoluciona la longitud:

- diseño de actividades didácticas donde el estudiante transforma representaciones y observa estabilización/variación de longitud,
- evaluación de protocolos (contar espacios, tildes, tokens) y su impacto en la dinámica,
- análisis de compresión semántica: una re-expresión puede acortar o alargar según la regla [36], [6].

III-F6. Bioinformática: codificación y compresión de secuencias: En bioinformática, secuencias (DNA/RNA/proteínas) se codifican y comprimen para almacenamiento y comparación. Si n representa longitud de secuencia o tamaño de archivo, α modela un esquema de codificación/compresión especializado y $S(n)$ el tamaño comprimido. Iterar el proceso permite analizar estabilidad del formato y overhead, y comparar codificadores por sus atractores (longitud estable) y tiempos de estabilización [37], [31].

III-F7. Aplicación en Ingeniería: dimensionamiento de redes y buffers en IIoT con re-encapsulación: En arquitecturas IIoT (sensores $\rightarrow$ gateway $\rightarrow$ broker $\rightarrow$ nube), un mensaje suele atravesar etapas de serialización, encapsulación (cabeceras/metadatos), compresión y cifrado autenticado. Cuando el pipeline se ejecuta repetidamente (reintentos, re-empaquetado, túneles, normalizaciones), el tamaño del mensaje induce naturalmente un SDS de longitud: el estado n_k es el tamaño (en bytes) antes del paso k, la codificación α representa el *proceso completo* y $S(n) = |\alpha(n)|$ es el tamaño final.

Un modelo operativo sencillo y realista es:

$$S(n) \;=\; B\left\lceil \frac{c\,(n+h)+t}{B} \right\rceil, \tag{8}$$

donde:

- h = overhead fijo (cabeceras, campos de longitud, metadatos),

Trayectorias del SDS de tamaño en pipeline IIoT (modelo (8))

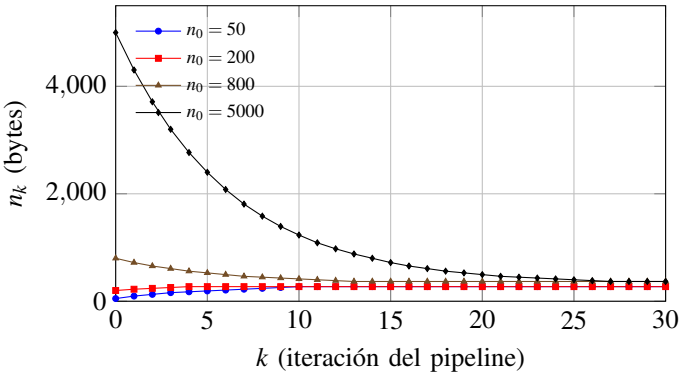

Figura 2. Simulación del SDS de tamaño de mensajes en un pipeline IIoT: al repetir encapsulación+compresión+cifrado, el tamaño converge a longitudes estables (atractores) que dependen del rango inicial y del alineamiento B.

- $c \in (0,1]$ = factor efectivo de compresión (cerca de 1 si "ya está comprimido"),
- t = bytes extra de autenticación (tag/MAC/checksum),
- B = alineamiento/padding (p.ej. múltiplos de 16 bytes).

Lectura ingenieril.

- Si $c < 1$, el sistema tiende a estabilizarse en longitudes "atractoras" (costos previsibles de transmisión).
- Si $c \approx 1$ y $h+t$ dominan, puede aparecer *inflación* por overhead al repetir el pipeline.
- La cuantización por B genera varios puntos fijos (longitudes estables) dependiendo del rango inicial.

Además, de (8) se obtiene una cota útil:

$$S(n) \le c(n+h)+t+B,$$

y por tanto si $c < 1$ existe un umbral N tal que para $n > N$ se cumple $S(n) < n$ (contracción), lo cual garantiza entrada a una región finita y estabilización eventual. En práctica, esto se usa para dimensionar buffers en gateways y para detectar pipelines que están re-procesando mensajes sin aportar beneficio (p.ej. recomprimir lo ya comprimido).

Comentario metodológico. Estas aplicaciones muestran que el SDS no es sólo un formalismo abstracto: funciona como una herramienta para predecir estabilización, inflación de tamaño y costos de re-procesamiento. Además, el marco de Teorema 1 (contracción) y Teorema 4 ($\log^*$) proporciona criterios verificables para asegurar convergencia rápida cuando se dispone de cotas logarítmicas sobre longitudes [29], [4].

Cuadro II
TRADUCCIÓN DEL SDS A ESCENARIOS REALES: ELECCIÓN DE α Y LECTURA DE $S(n) = |\alpha(n)|$.

Dominio	Interpretación de α y fenómeno
Compresión	$\alpha(n)$: comprimir objeto de tamaño n; estabiliza o crece por overhead [31], [32].
Redes	$\alpha(n)$: encapsular payload n en capas; crecimiento por headers/padding [34].
ETL/Logs	$\alpha(n)$: serializar+comprimir registro; inflación por metadatos repetidos [33].
Criptografía	$\alpha(n)$: hash/MAC; longitud fija $\Rightarrow$ atractor fuerte [35].
NLP/Educación	$\alpha(n)$: nombre/tokenización; sensibilidad a protocolo de conteo [36].
IIoT/Ingeniería	$\alpha(n)$: pipeline (serializar→encapsular→comprimir→cifrar) con reintentos; estabilización por cuantización/alineamiento de buffers [34], [31].

IV. DISCUSIÓN DE RESULTADOS

IV-A. De la universalidad a la teoría "con contenido"

La universalidad muestra que el SDS de longitud puede simular cualquier $F : \mathbb{N} \to \mathbb{N}_0$ si no se restringe α; por tanto, la agenda científica debe anclarse en *clases* de codificaciones con propiedades estructurales (inyectividad, compacidad, computabilidad, prefijo-libre, gramáticas regulares, etc.) [2], [5], [6]. En estas clases, los teoremas de contracción y potencial se vuelven operativos: basta controlar el crecimiento de la longitud para deducir existencia de atractores finitos. Esta lógica es análoga a la forma en que la dinámica simbólica obtiene resultados globales usando estructuras de desplazamiento y codificación [14], [15].

IV-B. Contracción en codificaciones naturales

En codificaciones posicionales, la contracción $S(n) = \Theta(\log n)$ es inmediata y conduce a atractores pequeños [10]. En códigos compactos, las longitudes controladas por entropía también favorecen contracción, aunque con un matiz: la longitud depende de una distribución o modelo, por lo que el análisis puede ser *en promedio* o *en el peor caso* [1], [2], [3]. En codificaciones lingüísticas, la evidencia heurística suele indicar crecimiento sublineal (por composicionalidad de nombres), pero la prueba formal requiere especificar gramática, convenciones y normalizaciones.

IV-C. Interpretación en grafos funcionales y cuencas

Una vez que se prueba atrapamiento en un conjunto finito B, el SDS se reduce a un problema finito: enumerar ciclos en el subgrafo inducido por B y describir cuencas [16]. Este enfoque es metodológicamente fuerte porque separa el trabajo en dos pasos: (i) prueba de contracción/potencial para garantizar B, y (ii) enumeración exacta de atractores dentro de B. En aplicaciones, el paso (ii) suele ser automatizable y reproducible.

IV-D. Limitaciones y sensibilidad

El SDS lingüístico es particularmente sensible al protocolo de conteo (espacios, tildes, signos), lo cual es un recordatorio de que la observable "longitud" no es universal, sino una función que debe definirse cuidadosamente [6]. Además, si Σ es numerable y no finito, las cotas combinatorias como (5) cambian cualitativamente; por ello, para resultados fuertes conviene trabajar con alfabetos finitos o imponer restricciones de complejidad a las palabras [8], [4]. Finalmente, aunque la contracción suele implicar atractores pequeños, pueden existir diseños de α que generen oscilaciones complejas; esta tensión es precisamente lo que justifica estudiar clases y no casos aislados [11].

IV-E. Alcance de la validación y líneas futuras

La evaluación computacional incorporada permite mostrar que el SDS de longitud no solo posee interés teórico, sino también utilidad práctica para analizar estabilización, inflación de mensajes y dimensionamiento preliminar de buffers en pipelines IIoT. Cuando la compresión domina el overhead, el sistema tiende a estabilizarse en longitudes atractoras; en cambio, cuando el mensaje es casi incompresible o ya fue procesado previamente, la repetición del pipeline puede producir inflación por acumulación de cabeceras, metadatos y padding.

No obstante, el alcance de esta validación debe entenderse como computacional y controlado. La validación con tráfico real de sensores, la comparación con protocolos específicos como MQTT, CoAP, HTTP o AMQP, y la implementación en plataformas industriales productivas constituyen líneas futuras necesarias para consolidar la aplicabilidad del modelo en escenarios IIoT reales.

IV-F. Conjeturas y líneas futuras

Proponemos un conjunto de conjeturas para SDS de longitud en codificaciones naturales:

Conjetura A (Atractor pequeño en codificaciones compactas). Si α satisface una cota superior logarítmica (6) (con constantes efectivas), entonces existe un conjunto B de tamaño universalmente pequeño (dependiente solo de A, B) que contiene todos los atractores.

Conjetura B (Convergencia tipo $\log^*$). Bajo (6), el tiempo $\tau_M(n_0)$ para alcanzar un umbral fijo M es $O(\log^* n_0)$ [4], [5].

Conjetura C (Robustez cualitativa frente a protocolos). Para codificaciones lingüísticas derivadas de una misma gramática, distintos protocolos π cambian detalles (períodos y cuencas) pero preservan la propiedad global de contracción eventual, de modo que la dinámica sigue siendo eventualmente periódica para todo n_0.

Estas conjeturas delimitan una "teoría alrededor del SDS": un conjunto de axiomas verificables para α que fuerzan fenómenos globales (atractores pequeños, convergencia rápida) y habilitan comparación entre codificaciones.

V. Conclusiones y Recomendaciones

V-A. Conclusiones

- El SDSde longitud $S(n) = |\alpha(n)|$ ofrece un marco unificador para estudiar iteraciones en $\mathbb{N}$ inducidas por codificaciones simbólicas, conectando grafos funcionales, dinámica simbólica y teoría de códigos [14], [2].
- Se establecieron resultados estructurales universales: acotación implica periodicidad eventual, y se presentaron criterios simples de contracción/expansión que determinan la existencia de atractores finitos o divergencia [12], [11].
- Para alfabetos finitos y codificaciones inyectivas, se derivó una cota inferior combinatoria de longitud, mostrando que la compacidad tiene límites informacionales [8], [2].
- En codificaciones naturales (digitales, compactas, lingüísticas), es plausible (y frecuentemente demostrable) la contracción eventual, lo cual sugiere atractores pequeños y convergencia rápida.

V-B. Recomendaciones

- Definir explícitamente el *protocolo de longitud* (espacios, tildes, normalizaciones) para garantizar reproducibilidad en SDS lingüísticos [6].
- Para desarrollar "teoría con contenido", fijar clases de codificaciones α con hipótesis verificables (p. ej. cota logarítmica, prefijo-libre, computabilidad) y estudiar invariantes: atractores, cuencas y tiempos de entrada [5], [3].
- Complementar pruebas por contracción con métodos de potencial tipo Lyapunov para capturar contracción irregular [13].
- Implementar exploración computacional sistemática con detección de ciclos (Floyd) y enumeración exacta en conjuntos atrapantes, reportando parámetros y rangos de muestreo [10], [16].

Referencias

[1] C. E. Shannon, "A mathematical theory of communication," *Bell System Technical Journal*, vol. 27, pp. 379–423, 623–656, 1948.

[2] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, 2nd ed. Wiley, 2006.

[3] D. J. C. MacKay, *Information Theory, Inference, and Learning Algorithms*. Cambridge Univ. Press, 2003.

[4] M. Li and P. Vitányi, *An Introduction to Kolmogorov Complexity and Its Applications*, 4th ed. Springer, 2019.

[5] M. Sipser, *Introduction to the Theory of Computation*, 3rd ed. Cengage, 2012.

[6] J. E. Hopcroft, R. Motwani, and J. D. Ullman, *Introduction to Automata Theory, Languages, and Computation*, 3rd ed. Pearson, 2006.

[7] N. Chomsky, *Syntactic Structures*. Mouton, 1957.

[8] M. Lothaire, *Combinatorics on Words*. Addison-Wesley, 1983.

[9] A. Salomaa and M. Soittola, *Automata-Theoretic Aspects of Formal Power Series*. Springer, 1978.

[10] D. E. Knuth, *The Art of Computer Programming*, Vol. 1: Fundamental Algorithms, 3rd ed. Addison-Wesley, 1997.

[11] R. L. Devaney, *An Introduction to Chaotic Dynamical Systems*, 2nd ed. Westview Press, 2003.

[12] K. T. Alligood, T. D. Sauer, and J. A. Yorke, *Chaos: An Introduction to Dynamical Systems*. Springer, 1996.

[13] A. Katok and B. Hasselblatt, *Introduction to the Modern Theory of Dynamical Systems*. Cambridge Univ. Press, 1995.

[14] D. Lind and B. Marcus, *An Introduction to Symbolic Dynamics and Coding*. Cambridge Univ. Press, 1995.

[15] B. Kitchens, *Symbolic Dynamics: One-Sided, Two-Sided and Countable State Markov Shifts*. Springer, 1998.

[16] P. Flajolet and A. Odlyzko, "Random mapping statistics," in *Advances in Cryptology—EUROCRYPT*, Springer, 1989, pp. 329–354.

[17] J. C. Lagarias, "The $3x+1$ problem and its generalizations," *American Mathematical Monthly*, vol. 92, no. 1, pp. 3–23, 1985.

[18] L. G. Kraft, "A device for quantizing, grouping, and coding amplitude modulated pulses," M.S. thesis, MIT, 1949.

[19] B. McMillan, "Two inequalities implied by unique decipherability," *IRE Trans. Information Theory*, vol. 2, no. 4, pp. 115–116, 1956.

[20] D. A. Huffman, "A method for the construction of minimum-redundancy codes," *Proceedings of the IRE*, vol. 40, no. 9, pp. 1098–1101, 1952.

[21] W. Feller, *An Introduction to Probability Theory and Its Applications*, Vol. 1, 3rd ed. Wiley, 1968.

[22] P. Billingsley, *Probability and Measure*, 3rd ed. Wiley, 1995.

[23] A. V. Aho, R. Sethi, and J. D. Ullman, *Compilers: Principles, Techniques, and Tools*. Addison-Wesley, 1986.

[24] K. Petersen, *Ergodic Theory*. Cambridge Univ. Press, 1983.

[25] P. Walters, *An Introduction to Ergodic Theory*. Springer, 1982.

[26] J. Berstel and D. Perrin, *Theory of Codes*. Academic Press, 1985.

[27] M. Morse and G. A. Hedlund, "Symbolic dynamics," *American Journal of Mathematics*, vol. 60, no. 4, pp. 815–866, 1938.

[28] W. Parry, "Symbolic dynamics and transformations of the unit interval," *Transactions of the American Mathematical Society*, vol. 122, pp. 368–378, 1966.

[29] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 3rd ed. MIT Press, 2009.

[30] D. P. Bertsekas, *Dynamic Programming and Optimal Control*, Vol. 1, 4th ed. Athena Scientific, 2017.

[31] K. Sayood, *Introduction to Data Compression*, 5th ed. Morgan Kaufmann, 2017.

[32] D. Salomon, *Data Compression: The Complete Reference*, 4th ed. Springer, 2007.

[33] I. H. Witten, A. Moffat, and T. C. Bell, *Managing Gigabytes: Compressing and Indexing Documents and Images*, 2nd ed. Morgan Kaufmann, 1999.

[34] J. F. Kurose and K. W. Ross, *Computer Networking: A Top-Down Approach*, 7th ed. Pearson, 2017.

[35] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*. CRC Press, 1996.

[36] D. Jurafsky and J. H. Martin, *Speech and Language Processing*, 3rd ed. draft. Prentice Hall / online manuscript, 2023.

[37] R. Durbin, S. R. Eddy, A. Krogh, and G. Mitchison, *Biological Sequence Analysis*. Cambridge Univ. Press, 1998.