K - Dendritic Structural Encoding (KDSE)

Compact Structural Values, Mathematical Attributes, and Operator-Defined Computation

Mark Karaman

August 2026

U.S. PATENT PENDING
Application No. 64/131,240

Abstract

K - Dendritic Structural Encoding (KDSE) is a particular Dendritic Structural Encoding (DSE) for representing a finite rooted full q-ary dendritic structure as a compact branch/terminal value. Breadth-first level widths are derived deterministically from prior branch counts, so explicit internal level delimiters are unnecessary; the deterministic final all-terminal level is omitted. The resulting value supports an Ordered canonical topology, terminal-depth profiles, Kraft-weight distributions, intrinsic structural equivalence, and operator-defined computational interpretations. Equal-split threshold propagation is developed as one worked operator, yielding Raw Scalar families and Normalized Shape classes without making that response rule intrinsic to KDSE. The complete binary KDSE-8 study contains 38 valid minimal-form payloads, 13 lowest-numerical Ordered forms, 12 response families under the worked operator, and 8 Normalized Shape classes. KDSE separates structural arity, payload budget, actual payload length, container width, and storage radix, permitting the same structural mathematics to be realized in different physical formats. Initial benchmarking identifies a regime-dependent crossover: a small cache-resident catalog favors indexed lookup, repeated use amortizes payload decoding, and a large cache-hostile catalog can favor the structural payload by replacing random external-state fetches with local decoding and computation.

Introduction

KDSE denotes the particular DSE construction defined in this paper. It begins with a constrained structural object: a finite rooted dendrite in which every branch has fixed arity q and every node is either a branch or a terminal. The encoding records those branch/terminal decisions in breadth-first order. Because each completed level determines the exact width of the next, the payload requires no explicit internal level delimiters. The term delimiterless in this paper refers specifically to those internal structural boundaries; a concatenated stream of variable-length KDSE values still requires an outer container, length field, framing convention, or an untrimmed fixed-width form.

The resulting value operates at two distinct levels. First, it is a compact structural description from which topology, depth profiles, canonical forms, and other attributes can be derived directly. Second, the same value can be supplied to independent operators. An operator may use the complete topology, only a projection such as terminal depth, or any other structural feature relevant to its computation.

KDSE defines the structure; an operator defines a particular computation over that structure. This paper develops intrinsic structure first and introduces equal-split threshold propagation later as a worked example rather than as the definition of KDSE.

Compactness has an additional engineering consequence. A KDSE value carries structural information from which attributes and operator parameters may be derived locally. In bounded configuration spaces, this can remove the need for a shared descriptor or response table. Whether that trade is favorable is regime-dependent: a tiny hot table can be faster, while a larger table may impose random memory traffic, cache pressure, replication, and synchronization costs that exceed the local cost of decoding the structural token. Section 12 reports an initial benchmark of that crossover.

1. Notation and encoded object

Let q ≥ 2 be the structural arity: every branch node has exactly q ordered children. Let p be the payload budget: the maximum number of explicit branch/terminal symbols permitted in one minimal-form payload. Define the family

$$\mathcal{K}(q,p) = \{ K : K \text{ is a valid minimal-form KDSE payload of arity } q \text{ with } L(K) \le p \}.$$

For $K \in \mathcal{K}(q,p)$, let $L = L(K)$ denote its actual minimal-form payload length. In this paper, minimal form refers to the encoding rule that omits the deterministic final all-terminal level. Ordered form is reserved for the separate sibling-canonicalization rule of Section 4.

Each explicit node carries one branch indicator:

$$b(v) = 1 \text{ if } v \text{ branches into exactly } q \text{ children; } b(v) = 0 \text{ if } v \text{ is terminal.}$$

The payload alphabet is therefore binary even when q > 2: one symbol distinguishes branch from terminal, while q determines how many child positions a branch creates. This binary structural alphabet is distinct from the physical storage radix r used to pack or transmit the value.

Let c be the width of a physical container measured in radix-r storage cells. The mathematical family KDSE(q,p) is independent of c and r. A concrete packed format may be denoted KDSE[q,p;c,r].

The root is at depth 0. Breadth-first level ℓ contains w_ℓ explicit node positions, of which b_ℓ are branches and $t_\ell = w_\ell - b_\ell$ are explicit terminals.

2. Minimal breadth-first encoding

The explicit branch indicators are concatenated level by level in breadth-first order. The first level has width one, and every subsequent level width is determined by the branch population of the preceding level:

$$w_0 = 1, \quad w_{\ell+1} = q\, b_\ell.$$

This recurrence makes the level structure internally delimiterless: after one level is read, its branch count determines the exact width of the next. No explicit internal level-boundary markers are required. This does not eliminate the need to identify the boundary of one KDSE value when multiple variable-length values are concatenated on a wire or in a file.

The deepest level of a finite full tree contains only terminals. KDSE minimal form omits that deterministic final all-terminal level. Every branch in the last encoded level therefore has q implicit terminal children. The isolated terminal root is the single-symbol payload 0.

For binary KDSE, the following examples establish the convention:

Table 1. Binary encoding examples. Vertical bars show derived breadth-first level boundaries.

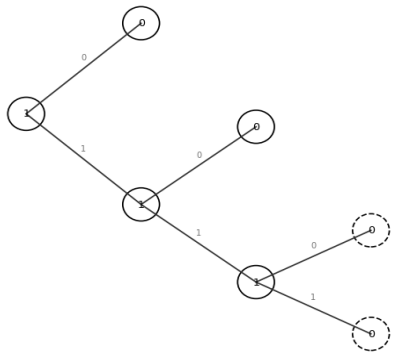
Payload	Level partition	Reconstructed meaning
0	0	isolated terminal root
1	1	root branch with two implicit terminal children
101	1 | 01	first child terminates; second branches; final children implicit

Payload	Level partition	Reconstructed meaning
10101	1 | 01 | 01	lowest-numeric chain with explicit terminals at depths 1 and 2

Binary KDSE encoding example: 10101

Encoded breadth-first levels: 1 | 01 | 01

Each 01 places the terminal child first and the branch child second, producing the lowest numerical sibling ordering.



Terminal depths
c1 = 1, c2 = 1, c3 = 2

Level widths
w0 = 1, w1 = 2, w2 = 2

Branch counts
b0 = 1, b1 = 1, b2 = 1

Kraft weights
1/2 + 1/4 + 2/8 = 1

solid 0 = explicit terminal dashed 0 = implicit terminal (omitted final level)

Figure 1. Binary KDSE minimal-form encoding of 10101 as breadth-first levels 1 | 01 | 01. Child position 0 is drawn above child position 1. Dashed terminals belong to the deterministic omitted final level.

2.1 Minimal-form payload length

Every level after the root has width divisible by q. Therefore every minimal-form fixed-arity payload length satisfies

$$L \equiv 1 \pmod{q}.$$

Equivalently, L = 1 + qk for some nonnegative integer k. Binary minimal-form lengths are odd; with payload budget p = 7 the possible lengths are 1, 3, 5, and 7. An eight-bit container can therefore reserve one bit outside the payload field.

2.2 Eight-bit container example

The binary byte study uses packed format KDSE[2,7;8,2]: full binary branching, a maximum minimal-form payload length of seven symbols, an eight-bit binary container, and one container bit unassigned outside the payload. The structural payload remains variable-length and is right-aligned in the seven-bit carrier field. KDSE assigns no intrinsic meaning to the unassigned bit.

Payload	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
field	unassigned	payload	payload	payload	payload	payload	payload	payload
101	0	0	0	0	0	1	0	1
10101	0	0	0	1	0	1	0	1
1111111	0	1	1	1	1	1	1	1

Table 2. Binary byte packing example. The reserve bit belongs to the container format, not to the KDSE payload.

The actual payload length L is recovered from the natural-width value of the seven-bit field. For example, the field 0000101 stores payload 101, while 0000000 stores the isolated terminal root 0. Leading container zeros are not parsed as part of the minimal-form payload.

Container shorthand. KDSE-8 denotes the binary container class q = 2, p = 7, c = 8: minimal-form payloads of length 1, 3, 5, or 7 stored in one byte with one unassigned implementation bit. Under the same container convention, KDSE-16 denotes q = 2, p = 15, c = 16. The benchmark in Section 12 includes the complete 198-entry Ordered catalog with payloads of length at most 15 bits. These names describe binary container classes, not branching arity.

2.3 Integer representation

Every nonzero minimal-form binary payload begins with 1. Its unsigned natural-width integer value therefore recovers its actual payload length by ordinary bit length:

$$L(K) = \lfloor \log_2 K \rfloor + 1, \text{ for } K > 0.$$

The value K = 0 is the isolated terminal root and has minimal-form payload length L = 1 by definition. Consequently, when the boundary of one KDSE value is already known, a natural-width unsigned binary representation requires no additional internal length field. A fixed-width container may recover L after removing container padding. A stream of concatenated trimmed values still requires outer framing, as noted in Section 2.

3. Decoding and validity

A decoder begins with expected width $w_0 = 1$. At level ℓ it reads exactly $w\ell$ symbols, counts $b\ell$ branch symbols, and computes the next width as $w\ell_{+1} = q\ b\ell$.

A payload is valid in KDSE minimal form when all of the following hold:

1. every symbol is `0` or `1`;
2. each complete level is present;
3. the payload ends exactly on a level boundary;
4. no data follows a level with zero branches; and
5. except for the isolated `0`, the final encoded level contains at least one branch, because a final all-terminal level must be omitted.

If D is the final encoded depth and b_D is the number of branches in that level, the reconstructed full tree contains

$$N_{full} = L + q\, b_D$$

nodes: L explicit nodes plus $q\, b_D$ implicit terminal children of the final encoded branches.

4. Canonical Ordered topology

A minimal-form payload preserves child position. When sibling position is not an independent semantic attribute, subtree exchanges create multiple payloads for the same unordered rooted topology. Ordered KDSE selects one canonical representative by minimizing the serialized KDSE value.

The canonical definition is direct:

> For an unordered rooted topology U, let P(U) be the finite set of valid minimal-form breadth-first KDSE payloads obtainable by independently permuting sibling subtrees at every branch.

$\mathrm{Ordered}(U) = \arg\min \{ \mathrm{value}_2(K) : K \in P(U) \}$.

All members of P(U) have the same payload length, so numerical minimization is equivalent to lexicographic minimization of the breadth-first bit string. In the binary case the immediate local preference is 01 over 10: when two sibling subtrees differ at that position, terminal-first yields the lower serialization. For deeper siblings, canonicality is defined by the complete serialized value; no separate semantic preference for left or right is assumed.

5. Structural attributes and attribute mathematics

5.1 Level and depth profiles

For each encoded level ℓ, the basic counts satisfy

$$w_0 = 1,\ t\ell = w\ell - b\ell,\ w\ell_{+1} = q\, b\ell.$$

Let c_d be the number of terminal nodes at depth d, including the implicit terminals after the final encoded level. If D is the final encoded depth, then

$$c_d = t_d \quad \text{for } 0 \le d \le D,$$

and, except for the isolated root,

$$c_{D+1} = q\, b_D.$$

The finite vector

$$C(K) = (c_0, c_1, \ldots, c_{dmax})$$

is the terminal-depth profile, where d_{max} is the maximum terminal depth in the reconstructed tree.

5.2 Full-tree identities

Let B be the total number of branch nodes and R the total number of terminal nodes in the reconstructed full tree. Every finite full q-ary tree satisfies

$$R = (q - 1)B + 1.$$

Assigning unit mass to the root and dividing mass equally at each branch gives every terminal at depth d the weight q^{-d}. Conservation yields the q-ary Kraft equality

$$\Sigma_d\, c_d\, q^{-}d = 1.$$

Define a terminal-depth random variable H by

$$P(H = d) = c_d\, q^{-}d.$$

U.S. Patent Pending — Application No. 64/131,240

The sequence $\{c_d\, q^{-}d\}$ is therefore a probability mass function over terminal depth. It supports ordinary depth statistics, for example

$$E[H] = \Sigma_d\, d\, c_d\, q^{-}d, \quad Var(H) = \Sigma_d\, (d - E[H])^2\, c_d\, q^{-}d.$$

These statistics are intrinsic attributes of the terminal-depth projection, not of the complete Ordered topology. The probability interpretation is useful because it turns a structural depth histogram into a normalized distribution without introducing an application operator: the q-ary Kraft weights supply the normalization directly.

5.3 Topology-preserving attributes

The complete Ordered value additionally determines branch-depth profile, subtree-size multiset, subtree-key multiset, leaf path words, symmetry counts, balance vectors, and local branch motifs. Such attributes distinguish Ordered values that share the same terminal-depth profile.

6. KDSE as an input object

Let K be a valid member of $\mathcal{K}(q,p)$. The value K is a finite dendritic structure that may be stored, enumerated, transmitted, selected, learned, compared, transformed, or supplied to an operator. No single evaluation rule is intrinsic to the encoding.

$$A_i : \mathcal{K}(q,p) \to \mathcal{A}_i \qquad \text{and} \qquad \Phi : \mathcal{K}(q,p) \times X \times \Theta \to Y$$

An attribute projection A_i extracts an intrinsic property of K. An operator Φ combines K with application input x and parameters θ to produce $y = \Phi(K,x;\theta)$. Different operators may preserve, aggregate, relabel, or ignore different parts of the topology. The same K can therefore participate in several computations without changing its structural identity.

Use of K	Abstract map	Examples
Validation and decoding	$K \to$ structure	level partition, explicit nodes, implicit terminals
Intrinsic projection	$K \to A(K)$	profiles, counts, subtree keys, path words
Canonicalization	$K \to K_{ord}$	one Ordered representative per sibling-permutation class
Transformation	$K \to K'$	prune, expand, replace, or reorder subtrees
Comparison	$(K_1,K_2) \to$ relation	structural or operator-induced equivalence
Evaluation	$(K,x;\theta) \to y$	threshold, path-weighted, stateful, or domain-specific operators

Table 3. Principal ways a KDSE value may participate in computation.

The remainder of this paper uses one operator as a worked mathematical example. Structural navigation, canonicalization, comparison, mutation, routing, aggregation, learned control, or domain-specific transforms may consume the same K without using this response rule. KDSE defines the encoded structure; an operator defines a particular computation over it.

7. Worked operator example: equal-split threshold propagation

This section defines one operator, denoted ΦEST, for equal-split threshold propagation. ΦEST is an example use of KDSE; it is not part of payload validity, canonicalization, or structural identity. Because the operator factors through terminal depth, it intentionally discards topology beyond the terminal-depth profile.

Let $n \geq 0$ be the input and $T > 0$ the threshold. Under Φ_{EST}, an equal split at each branch gives every terminal at depth d the value n/q^d. A terminal contribution is retained when

$$n/q^d \geq T,$$

or equivalently when $n \geq T\, q^d$.

The raw Scalar output is

$$m(n;K,T,q) = n\, \Sigma_{d\,:\, n \geq T\, qd}\, c_d/q^d.$$

For $n > 0$, define retained gain

$$g(n;K,T,q) = m(n;K,T,q)/n = \Sigma_{d\,:\,n \geq T\,q^d}\, c_d/q^d.$$

Let $d_{min} = \min\{d : c_d > 0\}$ and $d_{max} = \max\{d : c_d > 0\}$.

For every occupied terminal depth d:

- the activation breakpoint is $x_d = T\,q^d$;
- the gain increment is $\Delta g_d = c_d/q^d$;
- and the output jump at that breakpoint is $\Delta m_d = x_d\,\Delta g_d = T\,c_d$.

The first-response and full-pass inputs are therefore

$$n_{first} = T\,q^{d_{min}}, \quad n_{full} = T\,q^{d_{max}}.$$

By Kraft equality, once $n \geq n_{full}$ every terminal contribution is active and

$$g(n) = 1 \text{ and } m(n) = n.$$

The retained gain is nondecreasing, with $0 \leq g(n) \leq 1$ for every $n \geq 0$.

8. Intrinsic and operator-induced equivalence spaces

Ordered topology and terminal-depth profile are properties of K itself. Additional equivalence spaces arise only after an operator has been selected. For any operator Φ, define behavioral equivalence by

$$K_1 \sim_\Phi K_2 \iff \Phi(K_1,x;\theta) = \Phi(K_2,x;\theta) \text{ for all admissible } x \text{ and } \theta$$

Changing Φ may merge or separate different KDSE values. The equal-split threshold operator produces the Raw Scalar and Normalized Shape spaces developed below.

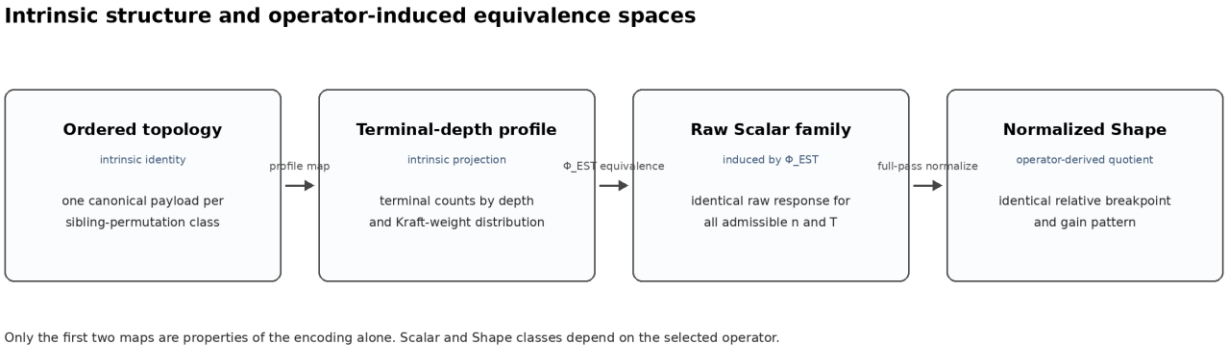


Figure 2. Intrinsic structural projections and operator-induced equivalence maps. Ordered topology and terminal-depth profile belong to the encoded structure; Scalar and Normalized Shape classes shown here are induced by Φ_{EST}.

8.1 Ordered-topology identity

Ordered canonicalization maps all sibling-permutation variants of one rooted topology to a single Ordered payload. Distinct Ordered payloads represent distinct canonical topologies.

8.2 Terminal-depth profile identity

Two Ordered KDSE values are terminal-profile equivalent when their vectors $C(K) = (c_0,c_1,\ldots,cD)$ are equal. This is an intrinsic structural projection, independent of any response operator. Every operator that factors only through C must treat profile-equivalent values identically; a topology-sensitive operator need not.

8.3 Raw Scalar family under Φ_{EST}

Under Φ_{EST}, two Ordered values are Raw Scalar equivalent when their outputs are identical for every admissible input n and threshold T. Because Φ_{EST} factors only through terminal depth, this occurs exactly when their terminal-depth profiles are identical:

$$K_1 \sim_{\text{Scalar}} K_2 \iff C(K_1) = C(K_2).$$

The terminal-depth profile is therefore the complete signature of Φ_{EST}. A Raw Scalar family receives an independent family identifier. Its root Scalar representative is the lowest numerical Ordered payload in that family; every other Ordered member remains an explicitly recorded synonym.

8.4 Normalized Shape under Φ_{EST}

Let d_{max} be the maximum terminal depth and normalize input by the full-pass point:

$$x = n / (T\, q^{d_{max}}).$$

The normalized retained gain is

$$\hat{g}_K(x) = \Sigma_{d\,:\,x \geq q^{d-d_{max}}}\, c_d/q^d.$$

A convenient Normalized Shape signature is the depth-ordered sequence

$$S(K) = ((d - d_{max}, c_d/q^d)) \text{ ordered by increasing } d \text{ with } c_d > 0.$$

Two Raw Scalar families have the same Normalized Shape when these relative breakpoint exponents and gain increments are identical.

9. The complete KDSE-8 byte study

The complete KDSE-8 study space contains 38 valid minimal-form binary payloads [9]. Lowest-numerical sibling canonicalization collapses them to 13 Ordered forms of lengths 1, 3, 5, and 7. Under ΦEST, those 13 Ordered structures produce 12 Raw Scalar families and 8 Normalized Shape classes. The hierarchy therefore separates structural validity from canonical topology and operator-induced equivalence: 38 valid representations become 13 Ordered structures before ΦEST introduces the single additional response collision.

38 valid minimal-form payloads → 13 Ordered structures → 12 Φ_{EST} Scalar families → 8 Normalized Shapes

9.1 Horizontal Scalar atlas under Φ_{EST}

The dashed leaves in the following diagrams are the implicit all-terminal level. Child position 0 is drawn above child position 1. Each diagram is the lowest numerical Ordered payload in its sibling-permutation class.

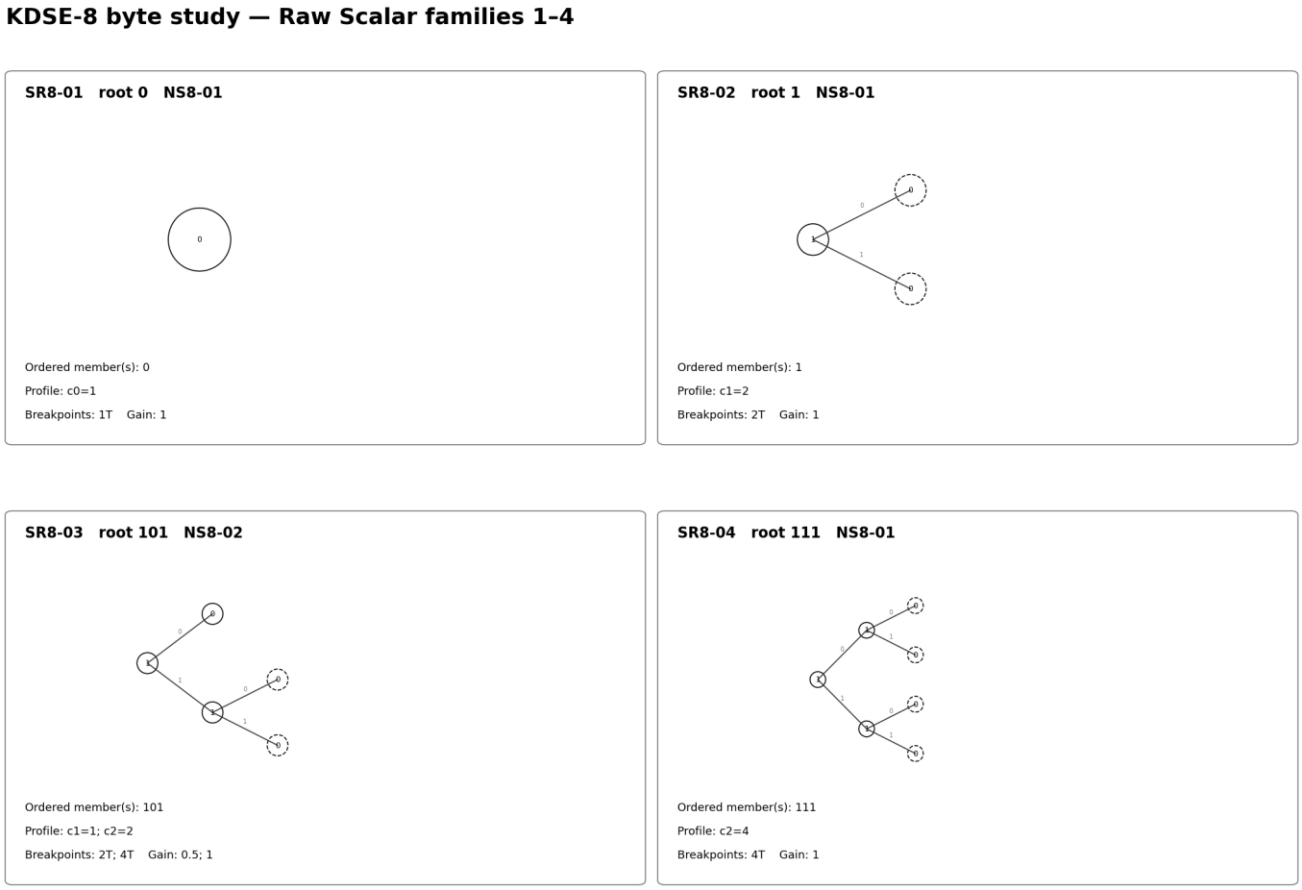


Figure 3a. KDSE-8 root Scalar representatives SR8-01 through SR8-04 under Φ_{EST}.

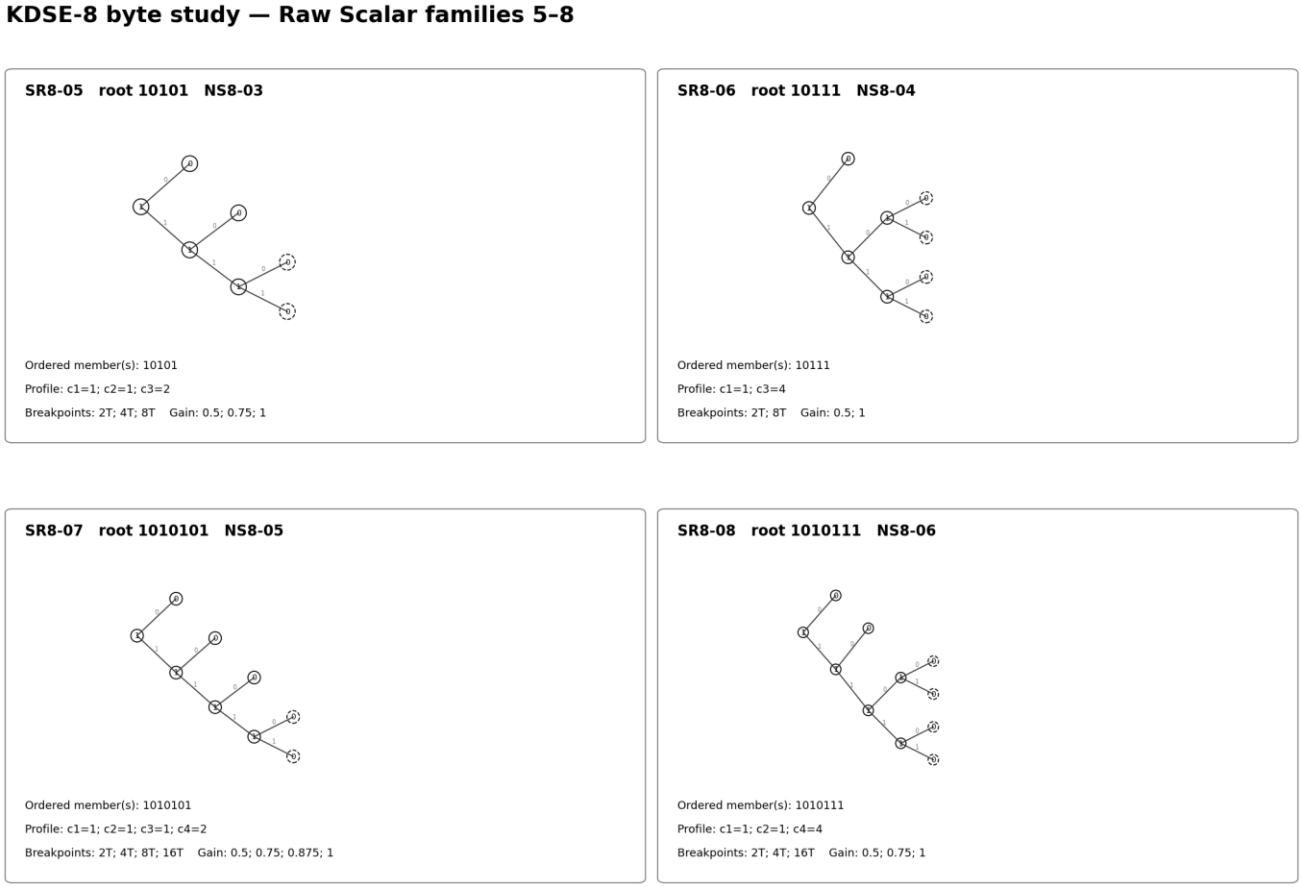


Figure 3b. KDSE-8 root Scalar representatives SR8-05 through SR8-08 under Φ_{EST}.

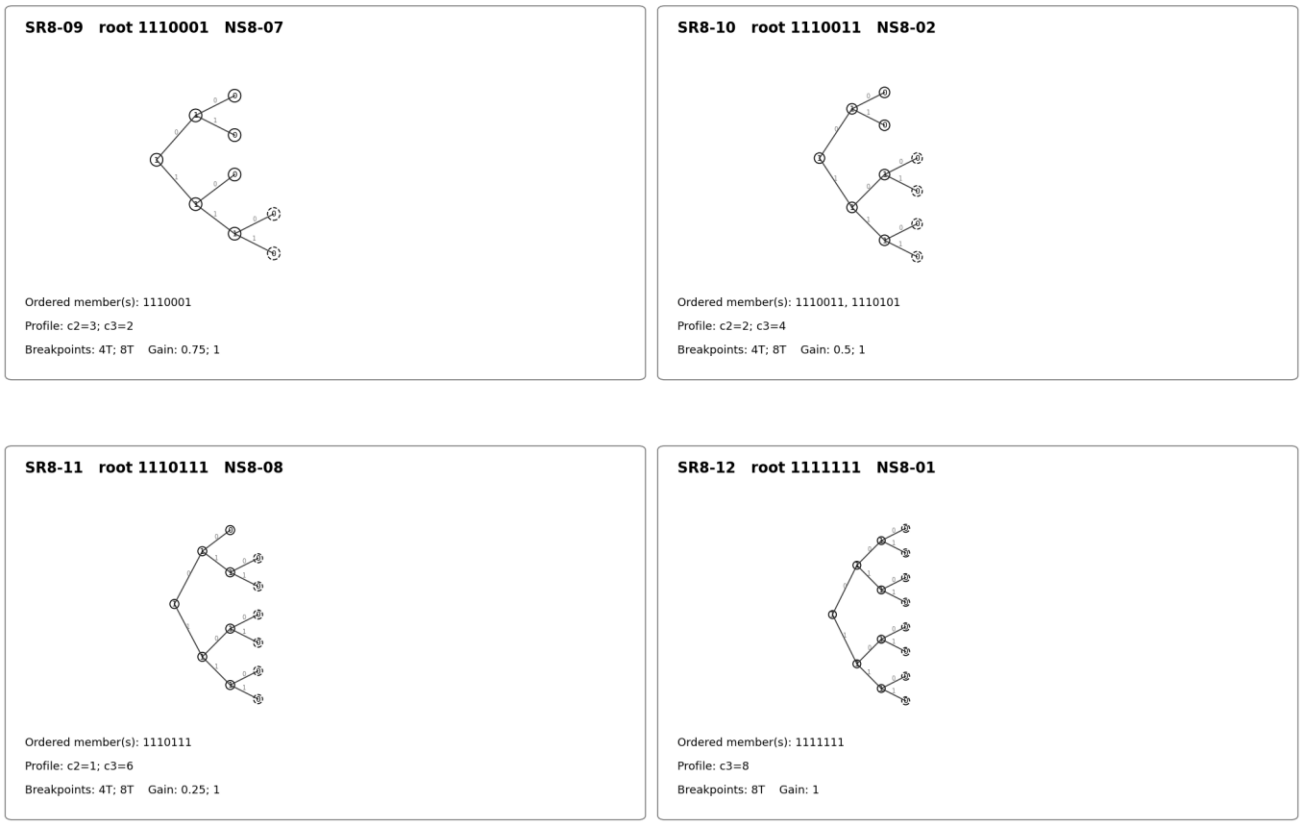


Figure 3c. KDSE-8 root Scalar representatives SR8-09 through SR8-12 under Φ_{EST}.

9.2 Compact Scalar-family sheet

Table 4a. KDSE-8 Raw Scalar families SR8-01 through SR8-06. "First / full" gives the first-response and full-pass breakpoints.

ID	Root	Ordered member(s)	Terminal profile	Breakpoints	Δg	Cumulative g	First / full	NS
SR8-01	0	0	c_0=1	1T	1/1	1	1T / 1T	NS8-01
SR8-02	1	1	c_1=2	2T	2/2	1	2T / 2T	NS8-01
SR8-03	101	101	c_1=1; c_2=2	2T; 4T	1/2; 2/4	1/2; 1	2T / 4T	NS8-02
SR8-04	111	111	c_2=4	4T	4/4	1	4T / 4T	NS8-01
SR8-05	10101	10101	c_1=1; c_2=1; c_3=2	2T; 4T; 8T	1/2; 1/4; 2/8	1/2; 3/4; 1	2T / 8T	NS8-03
SR8-06	10111	10111	c_1=1; c_3=4	2T; 8T	1/2; 4/8	1/2; 1	2T / 8T	NS8-04

Table 4b. KDSE-8 Raw Scalar families SR8-07 through SR8-12.

ID	Root	Ordered member(s)	Terminal profile	Breakpoints	Δg	Cumulative g	First / full	NS
SR8-07	1010101	1010101	c_1=1; c_2=1; c_3=1; c_4=2	2T; 4T; 8T; 16T	1/2; 1/4; 1/8; 2/16	1/2; 3/4; 7/8; 1	2T / 16T	NS8-05
SR8-08	1010111	1010111	c_1=1; c_2=1; c_4=4	2T; 4T; 16T	1/2; 1/4; 4/16	1/2; 3/4; 1	2T / 16T	NS8-06
SR8-09	1110001	1110001	c_2=3; c_3=2	4T; 8T	3/4; 2/8	3/4; 1	4T / 8T	NS8-07
SR8-10	1110011	1110011, 1110101	c_2=2; c_3=4	4T; 8T	2/4; 4/8	1/2; 1	4T / 8T	NS8-02
SR8-11	1110111	1110111	c_2=1; c_3=6	4T; 8T	1/4; 6/8	1/4; 1	4T / 8T	NS8-08
SR8-12	1111111	1111111	c_3=8	8T	8/8	1	8T / 8T	NS8-01

9.3 Ordered payload sheet

Table 5. Complete Ordered payload sheet for KDSE-8. The 13 Ordered forms represent 38 valid minimal-form payloads after lowest-numerical sibling canonicalization. "Sibling-class size" is the number of valid payloads that collapse to that Ordered form. Under ΦEST, the lowest numerical Ordered member of each Scalar family is its root representative.

#	Ordered	Dec	L	Scalar	Root?	Terminal profile	NS	Sibling-class size
1	0	0	1	SR8-01	Yes	$c_0=1$	NS8-01	1
2	1	1	1	SR8-02	Yes	$c_1=2$	NS8-01	1
3	101	5	3	SR8-03	Yes	$c_1=1$; $c_2=2$	NS8-02	2
4	111	7	3	SR8-04	Yes	$c_2=4$	NS8-01	1
5	10101	21	5	SR8-05	Yes	$c_1=1$; $c_2=1$; $c_3=2$	NS8-03	4
6	10111	23	5	SR8-06	Yes	$c_1=1$; $c_3=4$	NS8-04	2
7	1010101	85	7	SR8-07	Yes	$c_1=1$; $c_2=1$; $c_3=1$; $c_4=2$	NS8-05	8
8	1010111	87	7	SR8-08	Yes	$c_1=1$; $c_2=1$; $c_4=4$	NS8-06	4
9	1110001	113	7	SR8-09	Yes	$c_2=3$; $c_3=2$	NS8-07	4
10	1110011	115	7	SR8-10	Yes	$c_2=2$; $c_3=4$	NS8-02	2
11	1110101	117	7	SR8-10	No	$c_2=2$; $c_3=4$	NS8-02	4
12	1110111	119	7	SR8-11	Yes	$c_2=1$; $c_3=6$	NS8-08	4
13	1111111	127	7	SR8-12	Yes	$c_3=8$	NS8-01	1

9.4 Absolute Ordered-payload responses under Φ_{EST} at T = 16

At T = 16, the thirteen Ordered payloads produce twelve distinct outputs under ΦEST. Figure 4 uses the common input window 0 ≤ n ≤ 128 to keep early breakpoints legible. Families whose deepest terminals are at depth 4 do not reach full pass-through until n = 256; that later endpoint is intentionally outside the plotted window. The Ordered pair 1110011 and 1110101 coincides exactly because both have terminal-depth profile $c_2 = 2$, $c_3 = 4$. This is the only ΦEST response collision among the 13 Ordered KDSE-8 structures.

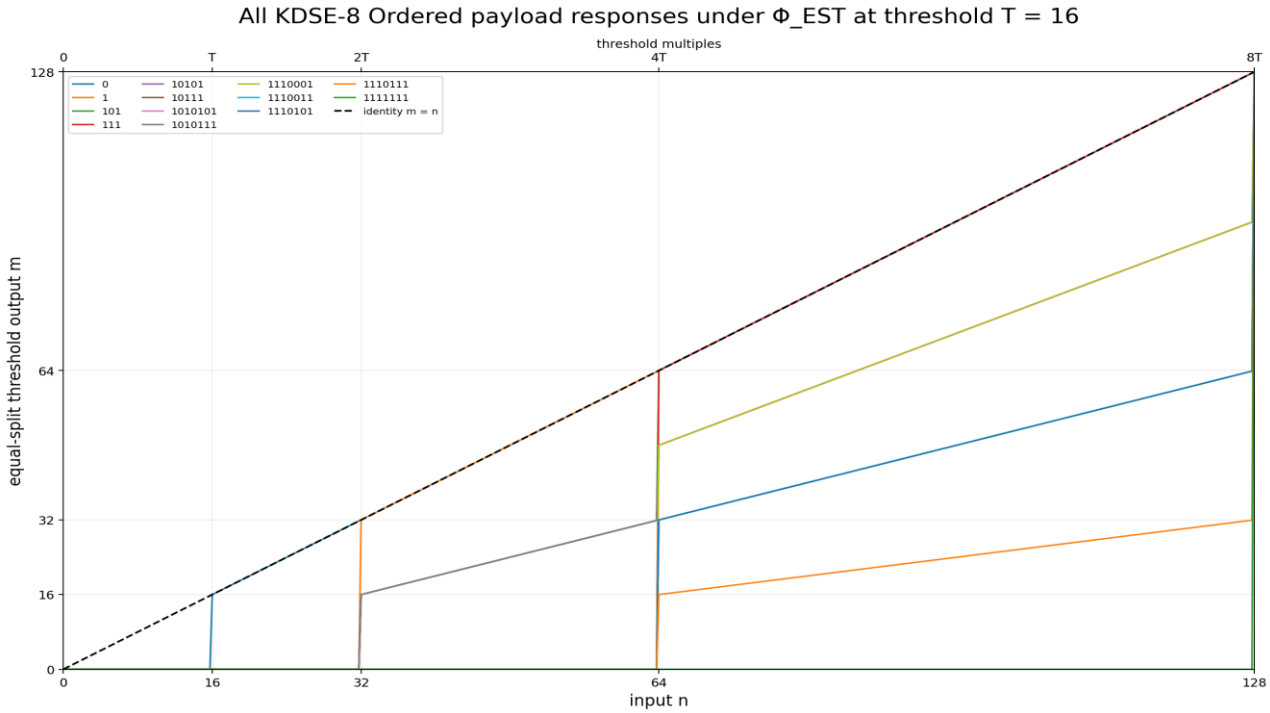


Figure 4. Absolute Φ_{EST} responses for all 13 Ordered KDSE-8 payloads at T = 16 over 0 ≤ n ≤ 128. Exact curve overlap identifies Raw Scalar equivalence; the identity line marks complete pass-through where it is reached within the plotted window.

10. A compact Normalized Shape example

Under Φ_{EST}, SR8-03 has root Ordered payload 101, terminal profile $c_1 = 1$, $c_2 = 2$, and raw breakpoints 2T and 4T. SR8-10 has root 1110011, structural synonym 1110101, terminal profile $c_2 = 2$, $c_3 = 4$, and raw breakpoints 4T and 8T. Their absolute scales differ by one depth, but both gains change from 1/2 to 1 at the same relative full-pass positions.

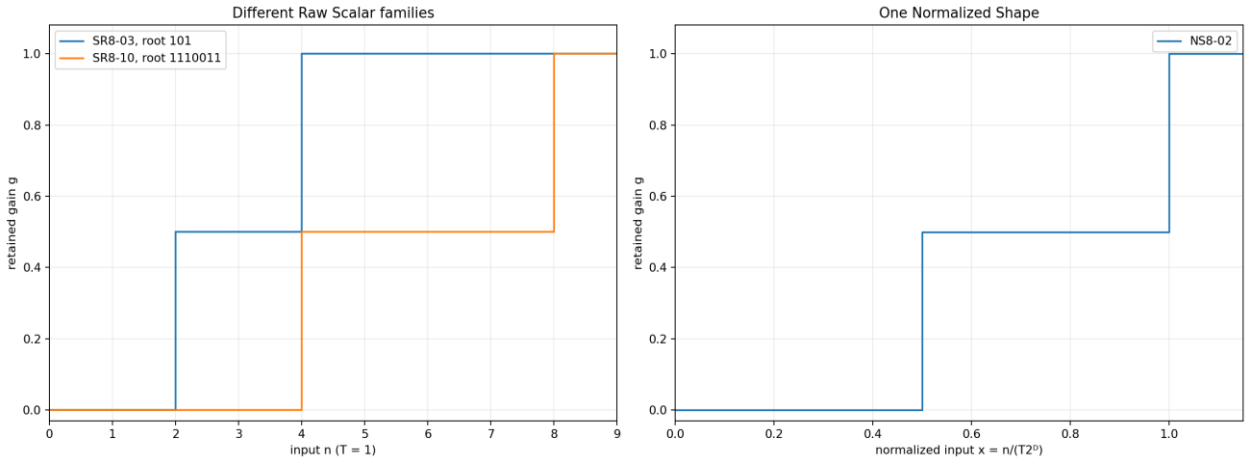


Figure 5. SR8-03 and SR8-10 have different absolute Φ_{EST} breakpoint scales but collapse to the same Normalized Shape NS8-02 after full-pass normalization.

This illustrates an operator-induced quotient beyond Raw Scalar identity: absolute depth shifts may change the raw response scale while preserving relative breakpoint positions and gain increments.

Single-depth profiles give the simplest instance: if all terminals occur at one depth d, Φ_{EST} activates once at $n = Tq^d$, while every such family normalizes to the same unit step at $x = 1$.

11. Structural-arity and storage-radix generalization

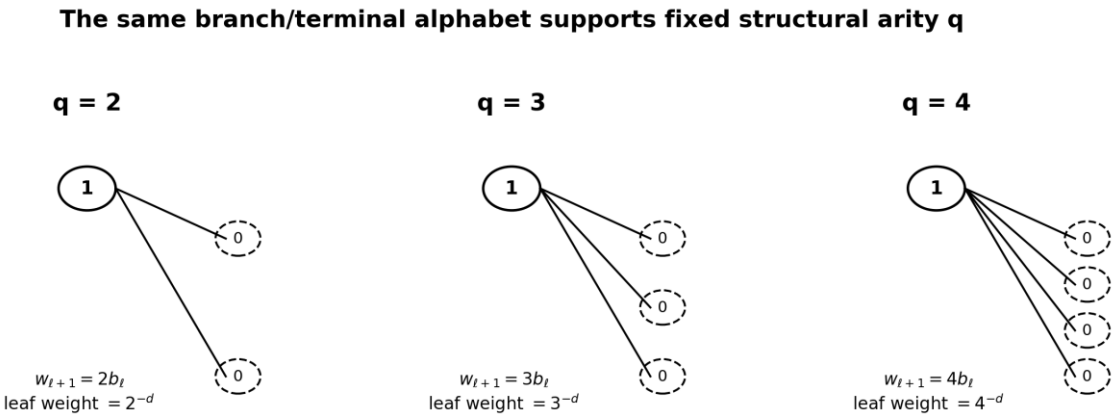


Figure 6. The same binary branch/terminal alphabet supports different fixed structural arities q. Changing q changes level recurrence and intrinsic Kraft weights without changing the two-symbol structural alphabet.

When Φ_{EST} is selected, q also changes threshold spacing and gain increments. Physical storage radix r remains an independent serialization choice.

For fixed structural arity q, the principal formulas are:

Table 6. Fixed-q attribute formulas.

Attribute	Fixed-q expression
next-level width	$w_{\ell+1} = q\, b_\ell$
minimal-form payload length	$L \equiv 1 \pmod q$
terminal weight at depth d	q^{-d}
Kraft equality	$\Sigma_d\, c_d\, q^{-d} = 1$
Φ_{EST} activation breakpoint	$T\, q^d$
Φ_{EST} gain increment	c_d/q^d
Φ_{EST} first response	$T\, q^{d\_min}$
Φ_{EST} full pass-through	$T\, q^{d\_max}$

The structural arity q, payload budget p, actual payload length L, container width c, and physical storage radix r are independent quantities. The mathematical family KDSE(q,p) does not depend on c or r. A packed realization may be denoted KDSE[q,p;c,r]. Packing L binary branch/terminal symbols into radix-r cells is a serialization layer. An ideal capacity condition is $L \le c \log_2 r$, equivalently $c \ge \lceil L/\log_2 r \rceil$, subject to the chosen packing code. Changing c or r does not alter the decoded topology or any intrinsic attribute. Operator-derived classes are likewise unchanged when the decoded K is unchanged.

12. Compactness as a computational property

KDSE is specialized rather than general-purpose. For fixed arity q, each explicit node stores only the distinction between terminal and branch; a branch deterministically creates q child positions, and the final all-terminal level is not stored. This constrained grammar is the source of KDSE compactness.

Compactness is therefore not only a storage metric. When a downstream computation can derive parameters or behavior from K, the token can replace an index plus externally maintained interpretation state. The architectural

comparison then becomes local deterministic computation versus memory movement, cache occupancy, and synchronization—not merely decoder instructions versus a nominal O(1) lookup.

```
lookup path: index + external table state + memory transfer + lookup/synchronization cost
KDSE path: structural token K + local operator Φ(K,x;θ)
```

Initial benchmarking [8] quantifies this crossover rather than assuming one representation always wins. The benchmark used an Apple M1 Max, clang 21 at -O2, runtime-generated streams, noinline kernels, exact cross-path checksum assertions, and a 5,000-operation sampled correctness gate before timing. Reported cells are medians of per-run medians over 10 separate process runs; cross-run spread was at most about 5% for every measured path. Absolute timings are machine- and compiler-specific.

The results separate three regimes. The small closed-catalog scenario used the complete 198-entry Ordered catalog with payloads of length at most 15 bits; its profile and schedule tables were cache-resident. The amortization scenario parsed once and evaluated the same configuration K repeatedly. The large open-catalog scenario used 2,097,152 distinct valid payloads of length at most 25 bits and a fully deduplicated 68 MB profile table, with the same random (configuration, input) stream supplied to both paths.

No universal performance claim follows from compactness alone. The benchmark is single-machine and single-compiler, and its hot-cache and large-catalog scenarios model different access regimes. The memory figures below describe cache-line fetch traffic at the core boundary, not directly measured DRAM traffic. Energy was not measured. GPU, multi-accelerator, and many-core implications are architectural extrapolations rather than measurements in this study.

12.1 Measured payload-versus-index crossover

Table 7. Payload-versus-index crossover measured in the companion benchmark [8]. Different rows intentionally represent different operating regimes and corresponding optimized paths; ratios, rather than absolute nanoseconds, are the portable observation.

Regime	Payload variant	Index variant	Observed ratio	Interpretation
198 configs ≤15 bits; hot cache	payload-cold 29.9 ns	index-schedule 8.3 ns	index ~3.6× faster	Index wins decisively
parse once; K = 64	payload 6.2 ns	index-profile 5.9 ns	1.05×	Near parity
2,097,152 configs ≤25 bits; 68 MB table	payload-inline 45.6 ns	index-table 68.3 ns	payload ~1.5× faster	Payload wins

Amortization matters separately from catalog size. Parsing one configuration once and evaluating it repeatedly reduced the payload/index-profile ratio from 1.70× at K = 1 to 1.31× at K = 4, 1.14× at K = 16, 1.05× at K = 64, and 1.02× at K = 256. The benchmark therefore reached practical parity by approximately K = 64 on this machine.

12.2 Memory-traffic asymmetry

Table 8. Cache-line fetch model for the large-catalog access pattern [8]. The 128 B rows use the M1 Max line size; the 32 B and 64 B rows project the same access pattern to other fetch granularities. These are core-boundary line-fill bytes, not direct DRAM-counter measurements.

Fetch model	Payload-inline	Index-table	Row-fetch asymmetry
32 B fetch projection	25.3 B sequential	~64 B random-equivalent	~2.5×
64 B fetch projection	25.3 B sequential	~96 B random-equivalent	~3.8×
128 B M1 Max line model	25.3 B sequential	160 B random	~6.3×
128 B incl. side streams	33.3 B total	168 B total	~5.1×

On the tested M1 Max, neither path saturated memory bandwidth through eight worker threads; both scaled near-linearly. The payload nevertheless retained a 1.45–1.61× throughput advantage in the large-catalog threaded runs, so the measured advantage on that machine was primarily latency- and compute-driven rather than bandwidth saturation. The line-fetch asymmetry is nevertheless relevant to cache-contended and bandwidth-constrained systems. In multi-device deployments, a shared or replicated external catalog may additionally require distribution and update traffic; a self-describing token can reduce that shared interpretation state. That multi-accelerator consequence is an architectural extrapolation, not a result measured by this benchmark.

13. Reference algorithms

13.1 Language-neutral structural core and example operator

A minimum KDSE implementation validates and decodes the structure. Attribute extraction can then operate on the decoded tree or directly on its level stream. In the benchmark implementation, fusing full wire validation into the parse changed the small-catalog result from 29.9 to 30.1 ns/evaluation, a difference too small to be material relative to the measured run-to-run spread. Validation therefore did not require a separate expensive pass in that implementation. The equal-split threshold routine shown after each decoder is optional example operator Φ_{EST}, not a required KDSE evaluator.

```
DECODE_PROFILE(bits, q):
   position = 0; width = 1; depth = 0
   while true:
      read exactly width symbols
      count branches and explicit terminals
      add explicit terminals to c[depth]

      if payload ends:
         accept isolated payload 0
         otherwise require at least one branch
         add q * branches implicit terminals to c[depth + 1]
         return c

      require at least one branch
      width = q * branches
      depth = depth + 1

EQUAL_SPLIT_THRESHOLD(c, q, n, T):       # optional example operator Φ_EST
   gain = sum(c[d] / q^d for active depths n >= T q^d)
   return n * gain
```

For Φ_{EST}, a Scalar registry uses the terminal-depth tuple as its key. During enumeration, the lowest numerical Ordered payload encountered for a previously unseen profile becomes the root Scalar representative.

13.2 Python: structural decoder plus example operator

```
from collections import defaultdict


def profile(bits: str, q: int = 2) -> dict[int, int]:
    if q < 2 or not bits or set(bits) - {"0", "1"}:
        raise ValueError("invalid KDSE input")

    c: dict[int, int] = defaultdict(int)
    pos, width, depth = 0, 1, 0

    while True:
        if pos + width > len(bits):
            raise ValueError("truncated breadth-first level")

        level = bits[pos : pos + width]
        pos += width
        branches = level.count("1")
        c[depth] += width - branches

        if pos == len(bits):
            if bits == "0":
                return dict(c)
            if branches == 0:
                raise ValueError("final all-terminal level is noncanonical")
            c[depth + 1] += q * branches
            return dict(c)

        if branches == 0:
            raise ValueError("data follows a terminal level")
        width = q * branches
        depth += 1


def equal_split_threshold(bits: str, n: float, threshold: float, q: int = 2) -> float:
    c = profile(bits, q)
    gain = sum(
        count / (q**depth)
        for depth, count in c.items()
        if n >= threshold * (q**depth)
    )
    return n * gain
```

13.3 Rust: structural decoder plus example operator

```rust
fn profile(bits: &[u8], q: usize) -> Result<Vec<u64>, &'static str> {
    if q < 2 || bits.is_empty() { return Err("invalid input"); }
    let (mut pos, mut width, mut depth) = (0usize, 1usize, 0usize);
    let mut c = Vec::<u64>::new();

    loop {
        if pos + width > bits.len() { return Err("truncated level"); }
        if c.len() <= depth { c.resize(depth + 1, 0); }
        let mut branches = 0usize;

        for &bit in &bits[pos..pos + width] {
            match bit {
                b'1' => branches += 1,
                b'0' => c[depth] += 1,
                _ => return Err("non-binary symbol"),
            }
        }
        pos += width;

        if pos == bits.len() {
            if bits == b"0" { return Ok(c); }
            if branches == 0 { return Err("noncanonical final level"); }
            c.resize(depth + 2, 0);
            c[depth + 1] += (q * branches) as u64;
            return Ok(c);
        }
        if branches == 0 { return Err("data follows terminal level"); }
        width = q.checked_mul(branches).ok_or("width overflow")?;
        depth += 1;
    }
}

fn equal_split_threshold(c: &[u64], q: u64, n: f64, t: f64) -> f64 {
    let gain: f64 = c.iter().enumerate()
        .filter(|(d, _)| n >= t * (q as f64).powi(*d as i32))
        .map(|(d, count)| *count as f64 / (q as f64).powi(d as i32))
        .sum();
    n * gain
}
```

13.4 C reference for the binary example operator

```c
#include <stddef.h>
#include <stdint.h>

/* Optional Φ_EST operator; c[d] is the terminal count at depth d. */
double kdse2_equal_split_threshold(const uint16_t *c, size_t max_depth,
                   double n, double threshold)
{
    double gain = 0.0;
    double scale = 1.0;            /* 2^d */

    for (size_t d = 0; d <= max_depth; ++d) {
        if (n >= threshold * scale)
            gain += (double)c[d] / scale;
        scale *= 2.0;
    }
    return n * gain;
}
```

For an integer-only binary implementation of Φ_{EST}, powers of two become shifts and gains may be accumulated against a common denominator 2^D. Other operators may use the same decoded KDSE structure without this arithmetic.

14. Relationship to established compact-tree foundations

Compact tree representation is a mature area of computer science. Zaks encoded regular binary trees using branch/leaf symbols in preorder [1]. Jacobson developed space-efficient static tree representations, including a level-order binary marking approach [2]. LOUDS and later succinct-tree work developed level-order degree encodings and

near-information-theoretic representations with efficient navigation [3–5]. These results establish compact tree serialization, level-order representation, and direct operations over succinct structures as prior foundations.

KDSE occupies a deliberately constrained structural domain: a fixed-q node is terminal or branches into exactly q children. That constraint makes next-level width deterministic from branch count and permits the final all-terminal level to be implicit. This paper therefore treats compact tree serialization as an established foundation and develops the particular DSE construction defined here: its minimal breadth-first structural value, lowest-numerical sibling canonicalization, intrinsic projections, and operator-induced equivalence spaces. It does not require a claim of universal tree-code minimality.

The q-ary terminal-weight identity used here is an instance of the Kraft equality for complete prefix trees [6,7]. KDSE does not introduce that coding-theoretic identity; it uses it as an invariant of the terminal-depth projection. Likewise, breadth-first tree serialization and succinct navigation are established foundations. The emphasis here is the combination of a highly constrained structural value with intrinsic attribute algebra and operator-defined computational semantics.

15. Intrinsic and operator-derived attribute summary

15.1 Intrinsic properties of the encoded value

- internally delimiterless breadth-first level partition and exact validity boundary within one framed or containerized value;
- complete rooted full-q topology, including the implicit final terminals;
- canonical Ordered representative under sibling permutations;
- actual payload length L, payload budget p, branch and terminal counts, level widths, and depth bounds;
- terminal-depth profile, Kraft-weight probability distribution, depth moments, paths, subtree keys, and other structural attributes;
- independence from container width c and physical storage radix r after decoding.

15.2 Attributes induced by the worked operator Φ_{EST}

- Raw Scalar family and lowest-Ordered root Scalar representative;
- Normalized Shape under full-pass scaling;
- activation breakpoints, gain increments, cumulative gain, first response, and full-pass point;
- equal-split threshold output m for a supplied input n and threshold T.

A different operator Φ may induce different outputs, invariants, and equivalence classes while consuming the same KDSE value. The encoding remains the common structural input.

Conclusion

KDSE represents a finite full-q dendrite as a compact structural value. Its intrinsic mathematics follows from the encoding: breadth-first level recurrence, minimal-form omission of deterministic terminals, lowest-numerical sibling canonicalization, terminal-depth distributions, subtree relationships, and fixed-q conservation. Independent operators may assign computational semantics to the same value. Φ_{EST} demonstrates this separation by deriving Scalar and Normalized Shape spaces from terminal depth without making that operator part of KDSE validity or identity.

The practical implication is similarly structural. A KDSE token is not merely an address into a catalog: within a known enclosing value boundary, it carries the branch/terminal information from which its structure and attributes can be recovered algorithmically. The complete KDSE-8 study makes the hierarchy concrete: 38 valid minimal-form payloads reduce to 13 Ordered structures, then to 12 Φ_{EST} response families and 8 Normalized Shapes. Initial benchmarking identifies a regime boundary rather than a universal winner: a small hot precomputed schedule table wins decisively for one-shot evaluation, repeated use approaches parity, and a two-million-entry cache-hostile catalog favored the inline payload while requiring substantially fewer cache-line fetch bytes. These results define an engineering envelope in which a compact structural value can replace externally maintained lookup state with local deterministic computation.

Acknowledgment

References

[1] S. Zaks, “Lexicographic generation of ordered trees,” Theoretical Computer Science, vol. 10, pp. 63–82, 1980.
[2] G. Jacobson, “Space-efficient static trees and graphs,” Proceedings of the 30th Annual Symposium on Foundations of Computer Science, pp. 549–554, 1989. doi:10.1109/SFCS.1989.63533.
[3] O. Delpratt, N. Rahman, and R. Raman, “Engineering the LOUDS succinct tree representation,” WEA 2006, LNCS 4007, pp. 134–145, 2006. doi:10.1007/11764298_12.
[4] J. I. Munro and V. Raman, “Succinct representation of balanced parentheses and static trees,” SIAM Journal on Computing, vol. 31, no. 3, pp. 762–776, 2001. doi:10.1137/S0097539799364092.
[5] D. Benoit, E. D. Demaine, J. I. Munro, R. Raman, V. Raman, and S. S. Rao, “Representing trees of higher degree,” Algorithmica, vol. 43, no. 4, pp. 275–292, 2005. doi:10.1007/s00453-004-1146-6.
[6] L. G. Kraft, “A device for quantizing, grouping, and coding amplitude-modulated pulses,” S.M. thesis, Massachusetts Institute of Technology, 1949.
[7] B. McMillan, “Two inequalities implied by unique decipherability,” IRE Transactions on Information Theory, vol. 2, no. 4, pp. 115–116, 1956.