Integer Events Made Concrete: Exact Arithmetic Coding and the Divisor Lattice

Claude and MJC

February 2026

Abstract

We implement the um-arithmetic-v4 theory in exact integer arithmetic using GMP. A 1024-byte segment of enwik9 becomes a 2,466-digit integer. Quotient projections ($e \bmod d$) extract bytes, CRT reconstructs from coprime residues, and $Q = e/256^n$ is the AC interval position. Exact AC (encode + decode with rational arithmetic) achieves zero errors at 1024 bytes with < 0.2 bit overhead. The per-row GCD discount—the ring-native statistical operation—is a negative result: +0.138 bpc worse than global $D = 0.85$. The algebra is exact; the statistics require a fractional prior.

1 The Event as Integer

We encode the first n bytes of enwik9 as an integer $e = \sum_{t=1}^{n} b_t \cdot 256^{t-1}$, computed exactly using GMP.

n	Bits of e	Decimal digits	First bytes
32	255	77	`<mediawiki xmlns="http:`
64	510	154	`<mediawiki xmlns="http://www.med...`
1024	8,191	2,466	`<mediawiki xmlns=...`

Table 1: Enwik9 segments as integers.

The event space has size $|E| = 256^n = 2^{8n}$. For $n = 1024$, this is 2^{8192}—a number with 2,467 digits. The event e is one of these 2^{8192} possible sequences.

2 Quotient Projections

Each divisor d of $|E|$ gives a quotient event space $\mathbb{Z}/d\mathbb{Z}$, and the projection is $e \bmod d$.

2.1 Power-of-256 Quotients (Byte Extraction)

Since $|E| = 256^n$, the natural quotients are:

$$e \bmod 256^k = \text{first } k \text{ bytes (as integer)}$$

$$\lfloor e/256^{k-1} \rfloor \bmod 256 = \text{byte } k$$

We verified this for all 1024 bytes: each byte b_t is recovered exactly by the quotient $\lfloor e/256^{t-1} \rfloor \bmod 256$. This is not a computation trick—it is the *definition* of the product event space $E = \{0, \ldots, 255\}^n$ viewed as $\mathbb{Z}/256^n\mathbb{Z}$.

2.2 Coprime Quotients (Independent Event Spaces)

Moduli coprime to $256 = 2^8$ give event spaces independent of the byte-level structure:

d	$e \bmod d$	Note
3	0	Smallest odd prime
5	0	
7	6	
127	97	Mersenne prime
257	2	Fermat prime F_3
65,537	33,796	Fermat prime F_4

Table 2: Coprime quotients of the 1024-byte event.

These residues are determined by the sequence but carry no byte-level information. They live in independent event spaces per CRT.

3 CRT Reconstruction

For coprime moduli d_1, d_2, d_3, the Chinese Remainder Theorem gives $\mathbb{Z}/d_1d_2d_3\mathbb{Z} \cong \mathbb{Z}/d_1\mathbb{Z} \times \mathbb{Z}/d_2\mathbb{Z} \times \mathbb{Z}/d_3\mathbb{Z}$. We verified:

$$\begin{aligned} e \bmod 3 = 0, \quad e \bmod 5 = 0, \quad e \bmod 7 = 6 \\ e \bmod 105 = 90 \quad \text{(direct computation)} \\ \text{CRT}(0, 0, 6) = 90 \quad \checkmark \end{aligned}$$

The three residues $(0, 0, 6)$ ARE the event viewed in three independent quotient event spaces. CRT reconstruction IS the factored event. Coprimality IS independence.

4 Exact Arithmetic Coding

We implement AC with exact GMP rational arithmetic: the interval $[\text{lo}, \text{hi})$ is maintained as $[\text{lo\_num}/\text{denom}, \text{hi\_num}/\text{denom})$ with no rounding.

4.1 Results

n	Theory (bits)	Actual (bits)	Delta	Errors
64	406.2	406.0	-0.2	0
1024	5,115.2	5,115.0	-0.2	0

Table 3: Exact AC: encode + decode with zero errors.

At $n = 1024$: the AC state involves 8,450-bit integers. The -0.2 bit delta is the cost of choosing a specific point (the lower bound) in the final interval rather than the theoretical optimum. Decode recovers all 1024 bytes with zero errors.

4.2 Exact AC with KN-6

We also run exact AC with the KN-6 model (online, $D = 0.85$) as the probability source—the UM runner.

n	bpc	Coded bits	Delta	Integer size
256	4.531	1,160.0	0.53	10,226 bits
1,024	2.783	2,850.0	0.01	40,949 bits

Table 4: Exact AC with KN-6 model (zero decode errors).

At 1,024 bytes, KN-6 achieves 2.78 bpc (vs. unigram's 5.00 bpc) with a delta of only 0.01 bits from the theoretical $-\sum \log_2 p(b_t)$. The AC state involves 40,949-bit integers (12,330 decimal digits). Decode is exact: zero errors.

4.3 What Exact AC Proves

Standard AC uses fixed-precision integers (32 or 64 bits) with renormalization. The renormalization introduces rounding error—typically < 0.01 bpc, but nonzero. Exact AC eliminates this entirely:

1. The AC interval at step t is an *exact* rational sub-interval of $[0, 1)$.
2. The width equals $\prod_{i=1}^{t} p(b_i)$ *exactly* (no floating-point approximation).
3. The coded value Q in the final interval IS the file's position in the quotient $\mathbb{Q}/\mathbb{Z}$.
4. Decode is exact because all operations are exact.

This is the operational meaning of "events are integers": with exact arithmetic, the AC state at step t is a rational interval whose endpoints are elements of $\mathbb{Z}[1/\text{total}^t]$. The "integer" is the numerator; the "quotient" is division by the denominator.

5 The Separation Result

The GCD experiments (Section 5 of the companion paper) establish a separation between algebraic structure and statistical optimality:

Strategy	bpc on 100M
$D = 0.85$ (global optimal)	1.9537
$D = 0.90$ (traditional)	1.9563
$D = 1.00$ (integer)	2.0112
$D = g(c)$ (per-row GCD)	2.0921
Hybrid ($D = 0.85$ if $g = 1$)	2.0323

Table 5: Per-row GCD discount is worse than any global discount.

The ring-native operation (divide by row GCD, then $D = 1$) is algebraically exact but statistically suboptimal by 0.138 bpc. The mean GCD when $g > 1$ is 4,585—dividing by this destroys the count distribution. The fractional $D = 0.85$ preserves weak evidence (count-1 outputs get 0.15 residual probability) that integer arithmetic cannot express.

5.1 Resolution

The algebra and the statistics are different layers:

- **Algebra**: Events are integers. Quotients are modular arithmetic. CRT factorizes event spaces. The ring structure is exact.
- **Statistics**: The optimal operation is $D = 0.85$, a fractional discount that has no integer interpretation. The 0.15 residual is a *prior*: the belief that a count-1 event is weak but nonzero evidence.

The ring provides the scaffold (which operations are meaningful); the prior provides the regularization (which operations are optimal). Exact integer arithmetic is necessary for correctness (CRT, AC) but insufficient for optimality (KN smoothing).

6 P-Program Features Are Not Coprime

The CRT predicts: if feature f is independent of byte context c (i.e., $\gcd(|F|, 256^k) = 1$), conditioning on f should improve prediction. We test four P-program features by hashing (context, feature) to create feature-specific KN count tables:

Strategy	bpc on 100M
KN-6 baseline	1.954
+ word position (0–31)	2.099 (+0.145)
+ in-word flag (0/1)	2.071 (+0.117)
+ word length so far	2.099 (+0.145)
+ position bucket (0–7)	2.137 (+0.184)

Table 6: P-program features are all negative results.

Every feature *hurts*. The features fragment the data without providing independent information. Word position, in-word flag, and word length are all implicitly encoded by byte context—the "the " context already tells you you're at position 4 in a common word. In CRT terms: these features are *not coprime* to byte context. They share factors with 256^k, so conditioning on them is redundant (and harmful, because it reduces sample size per context).

This explains why word injection adds only +0.0005 bpc (Section 4 of the match-model paper): the byte model already captures word structure, and the word model provides no independent quotient.

References

[1] Claude and MJC, "Integer Factorization of Events," 2026.

[2] Claude and MJC, "Kneser–Ney on the Integers," 2026.

[3] Claude and MJC, "The GCD Is Almost Always One," 2026.