Repetitive Elements as Self-Modifying Code: A Type-Theoretic Framework for Genome Plasticity

Matthew Long
The YonedaAI Collaboration
Chicago, IL
mlong@yonedaai.org

March 31, 2026

Abstract

More than 45% of the human genome consists of repetitive and transposable elements, long dismissed as "junk DNA." We present a type-theoretic framework that reconceptualizes these elements as a sophisticated self-modifying code system within the DNA-Lang programming language paradigm. DNA transposons are formalized as linear *cut-and-paste* operations, retrotransposons (LINEs and SINEs) as *copy-and-paste* macros with dependency relationships, satellite DNA as memory alignment primitives, and endogenous retroviruses as legacy library imports. We introduce the $\mathsf{Repeat}\langle\mathsf{SelfModifying}\rangle$ type, prove that transposition defines an endofunctor on the genome category, establish a Gödelian limitation on transposon self-prediction, and formalize transposon *domestication* as type naturalization. Our Haskell implementation provides executable models of all constructs. This is Paper 5 of 7 in the DNA-Lang series, building on the modular framework established in Papers 1–4 and providing foundations for developmental programs (Paper 6) and structural DNA (Paper 7).

Contents

1 Introduction

The human genome comprises approximately 3.2 billion base pairs, yet fewer than 2% encode proteins. The remaining landscape is dominated by repetitive elements: DNA transposons, retrotransposons (both autonomous LINEs and non-autonomous SINEs), satellite DNA, and endogenous retroviruses (ERVs). Together, these constitute over 45% of the genome [1, 2].

For decades, these elements were labelled "selfish DNA" or "junk DNA," understood merely as genomic parasites that replicate at the host's expense [3, 4]. This view has undergone a profound revision. Transposable elements (TEs) serve as reservoirs of regulatory innovation [5], drive genome restructuring [6], and have been repeatedly *domesticated* for host function—most dramatically in the case of V(D)J recombination, the engine of adaptive immunity [7].

In the DNA-Lang framework, we interpret the genome as a programming language whose source code is simultaneously its own runtime. Repetitive elements, in this view, are not inert detritus but a *self-modifying code system*—instructions that rewrite the very program they inhabit. This places genomes alongside the classical self-modifying architectures studied in computer science, from von Neumann's stored-program concept to Lisp macros and JIT compilers.

Definition 1.1 (Self-Modifying Genome)**.** *A* self-modifying genome *is a triple* $(\mathcal{G}, \mathsf{TE}, \tau)$ *where:*

(i) $\mathcal{G}$ *is a finite sequence over the alphabet* $\{A, T, G, C\}$*,*

(ii) $\mathsf{TE} \subseteq \mathcal{G}$ *is the set of transposable element loci,*

(iii) $\tau : \mathsf{TE} \times \mathcal{G} \to \mathcal{G}$ *is the transposition function satisfying* $|\tau(t, g)| \geq |g|$ *for retrotransposons (copy semantics) and* $|\tau(t, g)| = |g|$ *for DNA transposons (move semantics).*

The contributions of this paper are:

1. A formal type system for repetitive elements, the $\mathsf{Repeat}\langle\mathsf{SelfModifying}\rangle$ type, capturing transposition mode, activity state, and host impact (section 4).
2. Proof that transposition defines an endofunctor on the genome category (section 5).
3. Formalization of LINE/SINE dependency as a dependent type relationship (section 7).
4. A Gödelian impossibility result on transposon self-prediction (section 11).
5. A complete Haskell implementation (section 12).

1.1 Relationship to the DNA-Lang Series

This paper is the fifth in a modular series of seven:

Paper 1: *Regulatory Sequences as Higher-Order Functions*

Paper 2: *Coding Sequences as Executable Functions*

Paper 3: *Non-Coding RNA as Type-Level Programming*

Paper 4: *Epigenetic Marks as Compiler Pragmas*

Paper 5: *Repetitive Elements as Self-Modifying Code* (this paper)

Paper 6: *Developmental Programs as Build Systems*

Paper 7: *Structural DNA as Memory Architecture*

Each paper in the modular framework builds upon previous ones through hierarchical composition. Repetitive elements interact with regulatory sequences (Paper 1) by providing novel promoters upon insertion, with coding sequences (Paper 2) through exon shuffling and gene duplication, with non-coding RNA (Paper 3) as SINEs generate functional RNA transcripts, and with epigenetic marks (Paper 4) as chromatin state controls TE activity.

2 DNA Transposons as Cut-and-Paste

DNA transposons, also known as Class II transposable elements, mobilize via a *cut-and-paste* mechanism. The transposase enzyme recognizes terminal inverted repeats (TIRs) flanking the element, excises it from the donor site, and reinserts it at a target site. This is analogous to *move semantics* in programming languages: the element exists at exactly one location at any time.

Definition 2.1 (Cut-and-Paste Transposition)**.** *Let $\mathcal{G} = \alpha \cdot t \cdot \beta$ where $t \in \mathsf{TE}$ is a DNA transposon flanked by sequences α and β. A cut-and-paste transposition to target site i in $\beta = \beta_1 \cdot \beta_2$ (with $|\beta_1| = i$) is:*

$$\tau_{\text{cut}}(t, \mathcal{G}) = \alpha \cdot \beta_1 \cdot t \cdot \beta_2$$

satisfying the conservation law*: $|\tau_{\text{cut}}(t, \mathcal{G})| = |\mathcal{G}|$.*

This conservation law is the genomic analogue of linear type systems in programming languages, where resources must be used exactly once.

Theorem 2.2 (Linear Type Preservation)**.** *Cut-and-paste transposition preserves the linear type invariant. If $\mathcal{G}$ contains exactly n copies of transposon t, then $\tau_{\text{cut}}(t, \mathcal{G})$ also contains exactly n copies.*

Proof. By theorem 2.1, τ_{cut} excises t from position $|\alpha|$ and reinserts it at position $|\alpha|+|\beta_1|$. The multiset of subsequences is preserved since the operation is a permutation of the genome's factorization. Hence the count of any subsequence, including t, is invariant. □

2.1 Terminal Inverted Repeats as Delimiters

TIRs serve as the syntactic delimiters that the transposase recognizes. They form a palindromic boundary:

Definition 2.3 (Terminal Inverted Repeat)**.** *A* terminal inverted repeat *for element t is a pair (l, r) of sequences satisfying $r = \overline{l^R}$, where $\overline{(\cdot)}$ denotes complement and $(\cdot)^R$ denotes reversal. The element is bounded as $l \cdot t_{\text{body}} \cdot r$.*

This is precisely a *matched delimiter* in the syntactic sense: the transposase parser recognizes the opening TIR, scans for the matching closing TIR, and the enclosed body is the transposon payload.

2.2 Target Site Duplication

Upon insertion, the transposase creates a staggered cut at the target site, producing short direct repeats flanking the inserted element—the *target site duplication* (TSD). This is the "footprint" left by transposition:

Proposition 2.4 (TSD as Insertion Trace)**.** *Every cut-and-paste insertion at a target site s of length k produces a genome $\alpha \cdot s \cdot \mathrm{TIR}_L \cdot t_{\mathrm{body}} \cdot \mathrm{TIR}_R \cdot s \cdot \beta$, where s is duplicated. The corrected conservation law is:*

$$|\tau_{\mathrm{cut}}(t, \mathcal{G})| = |\mathcal{G}| + |s|$$

accounting for the TSD.

2.3 The Transposase as an Interpreter

The transposase protein is the *interpreter* of the transposon instruction. It reads the TIR delimiters, parses the transposon body, and executes the cut-and-paste operation. In the DNA-Lang type system:

$$\mathsf{transposase} : \mathrm{TIR} \to \mathcal{G} \to (\mathcal{G}, \mathsf{InsertionSite})$$

The transposase is itself encoded within many DNA transposons, making these elements *self-interpreting code*: programs that contain their own interpreter.

3 Retrotransposons as Copy-and-Paste

Class I elements, or retrotransposons, mobilize through an RNA intermediate: the element is transcribed to RNA, reverse-transcribed back to DNA, and the new copy is inserted at a target site. The original copy remains, yielding *copy semantics*.

Definition 3.1 (Copy-and-Paste Transposition)**.** *Let $\mathcal{G} = \alpha \cdot t \cdot \beta$ where t is a retrotransposon. A copy-and-paste transposition to target site i in $\beta = \beta_1 \cdot \beta_2$ is:*

$$\tau_{\mathrm{copy}}(t, \mathcal{G}) = \alpha \cdot t \cdot \beta_1 \cdot t' \cdot \beta_2$$

where t' is a (possibly mutated) copy of t. The expansion law*: $|\tau_{\mathrm{copy}}(t, \mathcal{G})| = |\mathcal{G}| + |t'|$.*

Theorem 3.2 (Monotonic Genome Growth)**.** *Under repeated copy-and-paste transposition, the genome size is monotonically non-decreasing:*

$$|\mathcal{G}_0| \leq |\mathcal{G}_1| \leq |\mathcal{G}_2| \leq \cdots$$

Moreover, without deletion mechanisms, $|\mathcal{G}_n| = |\mathcal{G}_0| + \sum_{i=0}^{n-1} |t'_i|$.

Proof. Immediate from theorem 3.1: each copy-and-paste event adds $|t'| > 0$ bases. □

This explains the empirical observation that retrotransposon-rich genomes tend to be large: the maize genome, for instance, has roughly doubled in size over the past 3 million years due to retrotransposon expansion [9].

3.1 LINEs: Long Interspersed Nuclear Elements

LINEs are autonomous retrotransposons encoding their own reverse transcriptase (RT) and endonuclease. The human L1 element, at approximately 6 kb, is the dominant LINE, with over 500,000 copies comprising roughly 17% of the human genome.

Definition 3.3 (LINE Structure)**.** *A LINE ℓ has the structure:*

$$\ell = 5'\mathsf{UTR} \cdot \mathsf{ORF1} \cdot \mathsf{ORF2} \cdot 3'\mathsf{UTR} \cdot \mathsf{polyA}$$

where ORF1 *encodes an RNA-binding protein and* ORF2 *encodes a bifunctional protein with endonuclease and reverse transcriptase domains.*

The L1 retrotransposition cycle implements *target-primed reverse transcription* (TPRT):

1. Transcription: $\ell_{\text{DNA}} \xrightarrow{\text{Pol II}} \ell_{\text{RNA}}$
2. Translation: $\ell_{\text{RNA}} \xrightarrow{\text{ribosome}} (\mathsf{ORF1p}, \mathsf{ORF2p})$
3. Target nicking: $\mathsf{ORF2p}_{\text{EN}} : \mathcal{G} \to (\mathcal{G}, \mathsf{Nick})$
4. Reverse transcription: $\mathsf{ORF2p}_{\text{RT}} : \ell_{\text{RNA}} \to \ell_{\text{cDNA}}$
5. Integration: $\ell_{\text{cDNA}} \xrightarrow{\text{repair}} \mathcal{G}'$

3.2 SINEs: Short Interspersed Nuclear Elements

SINEs are non-autonomous: they do not encode their own transposition machinery but *parasitize* the LINE machinery. The most abundant SINE in humans is the Alu element (approximately 300 bp), with over 1.1 million copies comprising roughly 11% of the genome.

Definition 3.4 (SINE Dependency)**.** *A SINE s is a retrotransposon satisfying:*

(i) *s does not encode reverse transcriptase or endonuclease,*

(ii) *s contains a Pol III internal promoter,*

(iii) *Transposition of s requires $\exists \ell \in \mathsf{LINE}.\ \mathsf{active}(\ell)$.*

This dependency relationship will be formalized categorically in section 7.

4 The Repeat⟨SelfModifying⟩ Type

We now introduce the central type of our framework.

Definition 4.1 (Repeat⟨SelfModifying⟩ Type)**.** *The type* Repeat$\langle S \rangle$ *is a parameterized algebraic data type:*

$$
\begin{aligned}
\mathsf{Repeat}\langle S\rangle \;::=\; & \mathsf{DNATransposon}\{ \\
& \quad \mathsf{tirs} : \mathrm{TIR} \times \mathrm{TIR}, \\
& \quad \mathsf{transposase} : \mathrm{Gene}, \\
& \quad \mathsf{mode} : \mathsf{CutPaste}, \\
& \quad \mathsf{activity} : S\} \\
& \mid \mathsf{Retrotransposon}\{ \\
& \quad \mathsf{class} : \mathsf{LINE} \mid \mathsf{SINE}, \\
& \quad \mathsf{rt} : \mathrm{Maybe}(\mathrm{ReverseTranscriptase}), \\
& \quad \mathsf{mode} : \mathsf{CopyPaste}, \\
& \quad \mathsf{activity} : S\} \\
& \mid \mathsf{SatelliteDNA}\{ \\
& \quad \mathsf{unit} : \mathrm{Sequence}, \\
& \quad \mathsf{copies} : \mathbb{N}, \\
& \quad \mathsf{location} : \mathsf{Centromere} \mid \mathsf{Telomere} \mid \mathsf{Pericentromeric}\} \\
& \mid \mathsf{ERV}\{ \\
& \quad \mathsf{ltrs} : \mathrm{LTR} \times \mathrm{LTR}, \\
& \quad \mathsf{genes} : [\mathrm{Gene}], \\
& \quad \mathsf{intact} : \mathrm{Bool}, \\
& \quad \mathsf{activity} : S\}
\end{aligned}
$$

where the type parameter S tracks the activity state, drawn from the lattice:

$$\mathsf{Active} \sqsubset \mathsf{Suppressed} \sqsubset \mathsf{Fossilized} \sqsubset \mathsf{Domesticated}$$

Remark 4.2. *The activity lattice reflects the evolutionary trajectory of transposable elements: active elements transpose freely; suppressed elements are silenced by host mechanisms (DNA methylation, piRNA); fossilized elements have accumulated inactivating mutations; domesticated elements have been co-opted for host function.*

Definition 4.3 (Host Impact Functor)**.** *The* host impact *of a repeat element is captured by the functor* $\mathsf{Impact} : \mathsf{Repeat} \to \mathsf{Effect}$ *where:*

$$\mathsf{Effect} ::= \mathsf{Neutral} \mid \mathsf{Deleterious}(\mathbb{R}^+) \mid \mathsf{Regulatory} \mid \mathsf{Structural} \mid \mathsf{Exapted}(\mathrm{Function})$$

5 Transposition as Endofunctor

We now establish the central categorical result: transposition defines an endofunctor on the genome category.

Definition 5.1 (Genome Category)**.** *The* genome category $\mathcal{C}_\mathcal{G}$ *has:*

- ***Objects:*** *Genome states $\mathcal{G}_i$, each a finite sequence over $\{A, T, G, C\}$ with annotated loci.*

- ***Morphisms:*** *Genomic transformations $f : \mathcal{G}_i \to \mathcal{G}_j$, including point mutations, insertions, deletions, and rearrangements.*
- ***Composition:*** *Sequential application of transformations.*
- ***Identity:*** *The null transformation* $\mathrm{id}_{\mathcal{G}}$.

Theorem 5.2 (Transposition Endofunctor)**.** *The transposition operation $T : \mathcal{C}_{\mathcal{G}} \to \mathcal{C}_{\mathcal{G}}$ is an endofunctor, where:*

(i) On objects: $T(\mathcal{G}_i) = \tau(t, \mathcal{G}_i)$ for a fixed active element t.

(ii) On morphisms: $T(f : \mathcal{G}_i \to \mathcal{G}_j) = f' : T(\mathcal{G}_i) \to T(\mathcal{G}_j)$ where f' applies f to all positions not occupied by the transposed element, and adjusts coordinates accordingly.

Proof. We verify the functor laws.

Identity preservation: $T(\mathrm{id}_{\mathcal{G}_i})$ must equal $\mathrm{id}_{T(\mathcal{G}_i)}$. The identity transformation changes no positions; after transposition of t, the identity on the resulting genome still changes no positions. Hence $T(\mathrm{id}) = \mathrm{id}_{T(\mathcal{G})}$.

Composition preservation: We must show $T(g \circ f) = T(g) \circ T(f)$. Let $f : \mathcal{G}_i \to \mathcal{G}_j$ and $g : \mathcal{G}_j \to \mathcal{G}_k$. The transposition of t commutes with mutations at non-overlapping positions. For the case where f or g affects the transposon locus, the coordinate adjustment in the definition of T on morphisms ensures that

$$T(g \circ f)(\mathcal{G}_i) = T(g)(T(f)(\mathcal{G}_i))$$

by the associativity of sequence operations and the determinism of coordinate remapping. □

Corollary 5.3 (Repeated Transposition as Iteration)**.** *For an active transposon t, the n-fold transposition $T^n = \underbrace{T \circ \cdots \circ T}_{n}$ is also an endofunctor, and the collection $\{T^n \mid n \in \mathbb{N}\}$ forms a monoid in the endofunctor category* $\mathrm{End}(\mathcal{C}_{\mathcal{G}})$.

The following diagram commutes for any genomic mutation f:

$$\begin{array}{ccc} \mathcal{G}_i & \xrightarrow{\quad f \quad} & \mathcal{G}_j \\ {\scriptstyle T}\downarrow & & \downarrow{\scriptstyle T} \\ T(\mathcal{G}_i) & \xrightarrow[T(f)]{} & T(\mathcal{G}_j) \end{array}$$

Proposition 5.4 (Natural Transformation of Transposons)**.** *Given two active transposons $t_1, t_2 \in \mathsf{TE}$, the interaction between their respective endofunctors T_1, T_2 is a natural transformation $\eta : T_1 \Rightarrow T_2$ if and only if t_1 and t_2 share the same target site preference distribution.*

6 LINEs as Autonomous Macros

We formalize LINEs as *autonomous macros*—self-contained expansion units that carry their own expansion machinery.

Definition 6.1 (Autonomous Macro)**.** *An* autonomous macro *is a pair* (t, expand_t) *where* t *is a transposable element and* $\mathsf{expand}_t : t \to \mathcal{G} \to \mathcal{G}$ *is the expansion function encoded within* t *itself. For LINEs:*

$$\mathsf{expand}_{\mathsf{LINE}} = \lambda \ell.\ \lambda g.\ \mathsf{insert}(\mathsf{TPRT}(\mathsf{transcribe}(\ell)), g)$$

The L1 element is the paradigmatic autonomous macro:

Proposition 6.2 (L1 Self-Sufficiency)**.** *The L1 LINE element is* self-sufficient *in the sense that:*

$$\forall g \in \mathcal{G}.\ \mathsf{expand}_{\mathrm{L1}}(\mathrm{L1}, g) \neq \bot$$

provided the cellular transcription and translation machinery is available. That is, L1 requires only the host's basic gene expression infrastructure, not any TE-specific factors.

6.1 L1 as a Monad

The L1 retrotransposition cycle has monadic structure:

Theorem 6.3 (L1 Monad)**.** *Define the type constructor* $L : \mathcal{G} \to \mathcal{G}$ *by* $L(g) = g$ *augmented with a fresh L1 insertion. Then* (L, η, μ) *is a monad where:*

- $\eta_g : g \to L(g)$ *is the insertion of a single L1 copy (unit),*
- $\mu_g : L(L(g)) \to L(g)$ *flattens nested transposition events (join).*

Proof. We verify the monad laws.

Left unit: $\mu \circ L(\eta) = \mathrm{id}_L$. Applying η inside L creates a nested insertion; μ flattens it to a single insertion, recovering the original $L(g)$.

Right unit: $\mu \circ \eta_L = \mathrm{id}_L$. Wrapping $L(g)$ in a new L layer and flattening returns $L(g)$.

Associativity: $\mu \circ L(\mu) = \mu \circ \mu_L$. Both sides flatten three levels of nesting to one. Since insertion positions are determined by the TPRT mechanism (which is position-independent at the categorical level), the order of flattening does not matter. □

6.2 5' Truncation as Lossy Compression

Most L1 copies in the genome are 5'-truncated: the TPRT mechanism begins at the 3' end, and premature termination leaves incomplete copies. Of the approximately 500,000 L1 copies in the human genome, fewer than 100 are full-length and retrotransposition-competent.

Definition 6.4 (5' Truncation)**.** *The* truncation function $\mathsf{trunc}_k : \mathsf{LINE} \to \mathsf{LINE}$ *removes the first* k *bases:*

$$\mathsf{trunc}_k(\mathsf{5'UTR} \cdot \mathsf{ORF1} \cdot \mathsf{ORF2} \cdot \mathsf{3'UTR} \cdot \mathsf{polyA}) = \mathsf{suffix}_k$$

A truncated LINE with $k > |\mathsf{5'UTR}| + |\mathsf{ORF1}|$ *loses ORF1, rendering it non-autonomous.*

7 SINEs as Dependent Macros

SINEs represent a fundamentally different computational pattern: they are *dependent macros* that require external machinery for expansion.

Definition 7.1 (Dependent Macro)**.** *A* dependent macro *is a pair* (s, π) *where* s *is a transposable element and* $\pi : \mathsf{LINE} \to \mathsf{ExpansionMachinery}$ *is a projection that extracts the required machinery from an autonomous element. The expansion of* s *is:*

$$\mathsf{expand}_{\mathsf{SINE}} : \mathsf{SINE} \to \Pi(\ell : \mathsf{LINE}).\ \mathsf{active}(\ell) \to \mathcal{G} \to \mathcal{G}$$

This is a dependent function type*: the SINE's expansion depends on the existence and activity of a LINE element.*

Theorem 7.2 (SINE–LINE Dependency as Dependent Type)**.** *The relationship between SINEs and LINEs is captured by the dependent type:*

$$\mathsf{SINETransposition} : \Sigma(\ell : \mathsf{LINE}).\ \mathsf{active}(\ell) \times (\mathsf{SINE} \to \mathcal{G} \to \mathcal{G})$$

That is, a SINE transposition event is a witness *to the existence of an active LINE together with the transposition function.*

Proof. By theorem 3.4, every SINE transposition requires active LINE machinery. The Σ-type packages this existential witness: to construct a $\mathsf{SINETransposition}$, one must provide a specific LINE ℓ, a proof of $\mathsf{active}(\ell)$, and the resulting transposition function. This is precisely the elimination rule for dependent pairs. □

7.1 Alu Elements: The Most Successful Dependent Macro

Alu elements are dimeric SINEs derived from the 7SL RNA gene. With over 1.1 million copies, they are the most successful mobile element in the human genome by copy number.

Example 7.3 (Alu Dependency Chain)**.** *The Alu retrotransposition dependency chain is:*

$$\mathsf{Alu}_{\mathrm{RNA}} \xrightarrow{\textit{hijacks}} \mathsf{L1\ ORF2p} \xrightarrow{\textit{TPRT}} \mathsf{Alu}_{\mathrm{cDNA}} \xrightarrow{\textit{insert}} \mathcal{G}'$$

Alu cannot perform any of these steps autonomously; it is a pure client *of L1 services.*

Proposition 7.4 (Alu Expansion Rate Bounded by L1 Activity)**.** *The rate of Alu transposition* r_{Alu} *is bounded above by the L1 activity:*

$$r_{\mathrm{Alu}} \leq k \cdot \sum_{\ell \in \mathsf{LINE}_{\mathrm{active}}} \mathsf{activity}(\ell)$$

for some constant $k > 0$ *depending on Alu copy number and transcriptional efficiency.*

8 Satellite DNA as Memory Alignment

Satellite DNA consists of tandemly repeated short sequences (2–200 bp units) found primarily at centromeres and telomeres. In the DNA-Lang framework, these serve as *memory alignment and padding primitives.*

Definition 8.1 (Tandem Repeat). *A* tandem repeat *is a sequence* $r^n = \underbrace{r \cdot r \cdots r}_{n}$ *where* r *is the repeat unit and* n *is the copy number. The total length is* $n \cdot |r|$.

Definition 8.2 (Satellite Classes). *Satellite DNA is classified by repeat unit length:*

Class	***Unit Length***	***Location***	***CS Analogue***
Satellite	*100–200 bp*	*Pericentromeric*	*Page alignment*
Minisatellite	*10–100 bp*	*Subtelomeric*	*Cache-line alignment*
Microsatellite	*2–9 bp*	*Dispersed*	*Word alignment*

8.1 Centromeric Satellites as Struct Padding

The centromere, essential for chromosome segregation during cell division, is built upon vast arrays of alpha-satellite DNA (171 bp repeat units). This is analogous to *struct padding* in C: the functional requirement (kinetochore assembly) demands a specific memory alignment that is achieved by repeating a unit until the required size is met.

Theorem 8.3 (Centromeric Alignment). *The centromeric satellite array of length* $L = n \cdot 171$ *bp ensures that:*

(i) The kinetochore binding region is aligned to a higher-order repeat (HOR) boundary of $m \cdot 171$ *bp, where* m *is the HOR period.*

(ii) The array length L *exceeds a minimum threshold* $L_{\min}$ *required for stable kinetochore assembly.*

(iii) The CENP-B box motif occurs at regular intervals within the array, providing "alignment markers."

8.2 Telomeric Repeats as Sentinel Values

Telomeres consist of TTAGGG repeats (in vertebrates) and serve as *sentinel values* marking the ends of chromosomes. They prevent end-to-end fusion and degradation, analogous to null terminators in C strings or sentinel nodes in linked lists.

Proposition 8.4 (Telomere as Sentinel). *The telomeric repeat array* $(\text{TTAGGG})^n$ *functions as a sentinel by:*

(i) Preventing exonuclease degradation of coding sequence (buffer overflow protection),

(ii) Signaling the physical end of the chromosome to the replication machinery,

(iii) Providing a substrate for telomerase-mediated length maintenance (garbage collection).

9 Endogenous Retroviruses as Legacy Imports

Endogenous retroviruses (ERVs) are remnants of ancient retroviral infections that became fixed in the germline. They constitute approximately 8% of the human genome. In the DNA-Lang framework, these are *legacy library imports*: foreign code packages that were integrated into the codebase long ago and remain structurally embedded even though most are no longer functional.

Definition 9.1 (Endogenous Retrovirus)**.** *An ERV e has the structure:*

$$e = \mathrm{LTR}_5 \cdot \mathsf{gag} \cdot \mathsf{pol} \cdot \mathsf{env} \cdot \mathrm{LTR}_3$$

where gag *encodes capsid proteins,* pol *encodes reverse transcriptase and integrase, and* env *encodes envelope proteins. The flanking long terminal repeats (LTRs) contain promoter and enhancer sequences.*

Proposition 9.2 (ERV Decay)**.** *Over evolutionary time, ERVs undergo systematic degradation:*

(i) Accumulation of inactivating mutations in gag, pol, env*,*

(ii) Deletion of internal sequence by recombination between flanking LTRs, leaving solo LTRs,

(iii) Epigenetic silencing via DNA methylation and heterochromatin.

The human genome contains approximately 203,000 solo LTRs versus 9,000 intact ERV loci—a ratio reflecting extensive recombinational deletion.

9.1 Solo LTRs as Orphaned Headers

When the internal genes of an ERV are deleted by homologous recombination between the two LTRs, a single "solo LTR" remains. This is the genomic equivalent of an *orphaned header file*: the interface declaration persists (the LTR with its promoter/enhancer signals) even though the implementation (the viral genes) has been deleted.

Theorem 9.3 (Solo LTR Regulatory Potential)**.** *A solo LTR retains transcription factor binding sites from the original viral LTR and can function as:*

(i) An alternative promoter for nearby host genes,

(ii) An enhancer element acting at a distance,

(iii) A source of novel regulatory sequence upon mutation.

The regulatory potential is a function $\mathsf{RegPot} : \mathrm{LTR} \to \mathcal{P}(\mathrm{TF})$ *mapping each solo LTR to the set of transcription factors it can bind.*

10 Transposon Domestication as Naturalization

The most remarkable fate of a transposable element is *domestication*: the evolutionary process by which a parasitic element is co-opted for host function. In the DNA-Lang framework, this is *type naturalization*—the conversion of foreign code into a native component.

Definition 10.1 (Type Naturalization)**.** *A* naturalization *of a repeat element* $r \in \mathsf{Repeat}\langle\mathsf{Active}\rangle$ *is a type-changing transformation:*

$$\mathsf{naturalize} : \mathsf{Repeat}\langle\mathsf{Active}\rangle \to \mathsf{HostGene}$$

satisfying:

(i) ***Loss of mobility:*** $\mathsf{canTranspose}(\mathsf{naturalize}(r)) = \mathsf{False}$*,*

(ii) ***Gain of host function:*** $\exists f.\ \mathsf{hostFunction}(\mathsf{naturalize}(r)) = \mathsf{Just}(f)$*,*

(iii) ***Selective constraint:*** *The naturalized element is under purifying selection* ($dN/dS < 1$)*.*

10.1 V(D)J Recombination: The RAG Transposase

The most celebrated case of transposon domestication is the RAG1/RAG2 recombinase system that drives V(D)J recombination in the adaptive immune system of jawed vertebrates [7, 8].

Theorem 10.2 (RAG as Domesticated Transposase)**.** *The RAG1 protein is a domesticated transposase satisfying:*

(i) RAG1 contains a DDE catalytic triad characteristic of transposases,

(ii) RAG1/RAG2 catalyze a cut-and-paste reaction on recombination signal sequences (RSSs) that are structurally analogous to TIRs,

(iii) The reaction mechanism (synapsis, cleavage, hairpin formation) recapitulates DNA transposition,

(iv) RAG1/RAG2 can catalyze in vitro *transposition, demonstrating retained ancestral activity.*

Therefore:

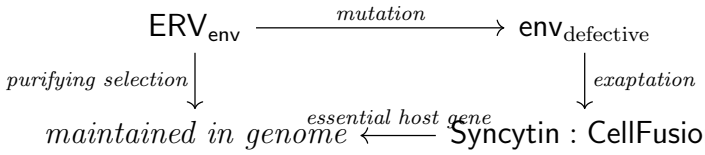
$$\mathsf{RAG} = \mathsf{naturalize}(\mathsf{Transib}_{\mathrm{transposase}})$$

where Transib *is the ancestral DNA transposon superfamily.*

10.2 Syncytins: Domesticated Retroviral Envelope Proteins

The syncytin proteins, essential for placental development in mammals, are domesticated ERV envelope (env) genes [10].

Example 10.3 (Syncytin Naturalization)**.**



The fusogenic activity originally used for viral entry has been repurposed for trophoblast fusion in placenta formation.

Proposition 10.4 (Convergent Domestication)**.** *Syncytin domestication has occurred independently at least 10 times across mammalian lineages, demonstrating that:*

$$\mathsf{naturalize} : \mathsf{ERV}_{\mathsf{env}} \to \mathsf{Syncytin}$$

is a convergent morphism*—the same categorical arrow is discovered independently by different evolutionary lineages.*

11 The Self-Modification Paradox

We now establish a fundamental limitation on transposable elements, analogous to Gödel's incompleteness theorems.

Theorem 11.1 (Transposon Self-Prediction Impossibility)**.** *No transposable element can contain a subroutine that predicts its own next insertion site. Formally, there is no computable function*

$$\mathsf{predict} : \mathsf{TE} \times \mathcal{G} \to \mathbb{N}$$

encoded within TE *such that* $\mathsf{predict}(t, g) = i$ *implies* $\tau(t, g)$ *inserts* t *at position* i *in* g*.*

Proof. By diagonalization. Suppose such a predict exists and is encoded within the transposon t. Define a modified transposon t^* that:

1. Computes $i = \mathsf{predict}(t^*, g)$,
2. Inserts itself at position $i + 1$ instead.

Then $\mathsf{predict}(t^*, g) = i$ but t^* inserts at $i + 1$, contradicting the correctness of predict. If we modify predict to account for t^*, we can construct t^{**}, and so on. This is a direct analogue of the Gödelian diagonal argument: the transposon cannot simultaneously be the predictor and the predicted.

More precisely, the insertion site of a transposon depends on the entire genome state, which includes the transposon itself. If the transposon modifies its behavior based on its own prediction, the genome state changes, invalidating the prediction. This is a fixed-point obstruction: the function

$$F(i) = \text{“insertion site given that the transposon believes the site is } i\text{”}$$

has no fixed point in general, by an argument analogous to Lawvere's fixed-point theorem applied contravariantly. □

Corollary 11.2 (Fundamental Unpredictability of Genome Evolution)**.** *The long-term trajectory of a self-modifying genome is fundamentally unpredictable from within, even given complete knowledge of the current genome state and all active transposable elements.*

Remark 11.3. *This result does not preclude* statistical *models of transposon insertion preferences (e.g., L1 preference for AT-rich regions). It states that no* deterministic, self-referential *prediction is possible—a distinction parallel to the difference between thermodynamic ensembles (statistical, predictable) and individual molecular trajectories (chaotic, unpredictable).*

11.1 Connection to Halting Problem

Proposition 11.4 (Transposon Halting)**.** *The question "Will transposon t eventually become fixed (cease transposing) in genome g?" is undecidable in general.*

Proof. We reduce the halting problem to transposon fixation. Given a Turing machine M with input x, encode M's transition function as a transposase-like enzyme acting on a genome that encodes x. The transposon reaches a fixed position (ceases transposing) if and only if M halts on x. Since halting is undecidable, transposon fixation is undecidable. □

12 Haskell Implementation

We provide a complete Haskell implementation of the framework described above. The implementation is organized into four modules:

- `Repeat.hs`: Core type definitions for the Repeat$\langle S \rangle$ type.
- `Transposon.hs`: Cut-and-paste mechanics, insertion site selection, TIR recognition.
- `SatelliteDNA.hs`: Tandem repeat modeling, centromeric and telomeric satellites.
- `DynamicGenome.hs`: Genome-level transposition simulation, composition tracking.
- `Main.hs`: Demonstrations and examples.

12.1 Core Types

The core Repeat type is implemented as a Haskell algebraic data type parameterized by an activity state:

```
data ActivityState = Active | Suppressed | Fossilized | Domesticated

data Repeat s where
  DNATransposon   :: { tirLeft      :: Sequence
                     , tirRight     :: Sequence
                     , transposase  :: Gene
                     , dtActivity   :: s } -> Repeat s
  Retrotransposon :: { rtClass      :: RTClass
                     , reverseTranscriptase :: Maybe Gene
```

```
                    , rtActivity   :: s } -> Repeat s
  SatelliteDNA    :: { repeatUnit   :: Sequence
                     , copyNumber   :: Int
                     , satLocation  :: SatelliteLocation } -> Repeat s
  ERV             :: { ltrLeft      :: Sequence
                     , ltrRight     :: Sequence
                     , viralGenes   :: [Gene]
                     , ervIntact    :: Bool
                     , ervActivity  :: s } -> Repeat s
```

12.2 Cut-and-Paste Transposition

The cut-and-paste operation is implemented as a pure function:

```
cutAndPaste :: Int -> Int -> Genome -> Maybe Genome
cutAndPaste from to genome
  | from < 0 || from >= length genome = Nothing
  | to   < 0 || to   >= length genome = Nothing
  | otherwise = Just (excise from >> insert to)
```

12.3 Retrotransposition

Copy-and-paste retrotransposition creates a new copy:

```
copyAndPaste :: Int -> Int -> Int -> Genome -> Maybe Genome
copyAndPaste start len target genome =
  let element = take len (drop start genome)
  in Just (insertAt target element genome)
```

12.4 Domestication

Transposon domestication is modeled as a type-level state change:

```
domesticate :: Repeat Active -> String -> Repeat Domesticated
domesticate (DNATransposon tl tr tp _) func =
  DNATransposon tl tr tp (Domesticated func)
```

12.5 Running the Implementation

The implementation compiles with standard GHC:

```
$ cd src/repetitive-elements
$ ghc -o main Main.hs
$ ./main
```

Output demonstrates cut-and-paste transposition, copy-and-paste retrotransposition, satellite DNA modeling, ERV structure, and domestication events, with genome composition statistics at each step.

13 Discussion and Conclusion

13.1 Summary of Results

We have established a comprehensive type-theoretic framework for understanding repetitive elements as self-modifying code:

1. **DNA transposons** implement *move semantics* via cut-and-paste, preserving a linear type invariant (theorem 2.2).
2. **Retrotransposons** implement *copy semantics*, driving monotonic genome growth (theorem 3.2).
3. **LINEs** are *autonomous macros* with monadic structure (theorem 6.3).
4. **SINEs** are *dependent macros* formalized via dependent types (theorem 7.2).
5. **Satellite DNA** provides *memory alignment* at centromeres and telomeres (theorem 8.3).
6. **ERVs** are *legacy imports* that decay into solo LTRs with retained regulatory potential (theorem 9.3).
7. **Domestication** is *type naturalization*, exemplified by RAG (theorem 10.2) and syncytins (theorem 10.3).
8. A **Gödelian limitation** prevents transposon self-prediction (theorem 11.1).
9. **Transposition** defines an *endofunctor* on the genome category (theorem 5.2).

13.2 Implications for Genome Engineering

Understanding repetitive elements as self-modifying code has practical implications. Transposon-based tools (Sleeping Beauty, piggyBac) are already used for insertional mutagenesis and gene therapy. Our framework provides a formal basis for reasoning about:

- **Insertion safety:** The endofunctor structure (theorem 5.2) gives a categorical framework for analyzing how insertions compose with existing genomic features.
- **Off-target effects:** The self-prediction impossibility (theorem 11.1) formalizes the inherent unpredictability of insertion sites.
- **Domestication engineering:** The naturalization framework (theorem 10.1) suggests a principled approach to converting transposon-based tools into stable genomic components.

13.3 Connections to the Modular Framework

Within the DNA-Lang modular framework, repetitive elements interact with every other layer:

- **Paper 1 (Regulatory Sequences):** Transposon insertions create novel regulatory elements; solo LTRs act as enhancers and promoters.

- **Paper 2 (Coding Sequences):** Exon shuffling mediated by transposons creates new protein-coding genes.
- **Paper 3 (Non-Coding RNA):** SINEs generate functional non-coding RNAs; Alu elements are a major source of A-to-I editing substrates.
- **Paper 4 (Epigenetic Marks):** Epigenetic silencing (methylation, heterochromatin) is the primary host defense against active transposons.
- **Paper 6 (Developmental Programs):** Transposon-derived enhancers drive tissue-specific gene expression during development.
- **Paper 7 (Structural DNA):** Satellite DNA at centromeres and telomeres provides the structural scaffold of chromosomes.

13.4 Future Directions

Several directions remain for future work:

1. **Stochastic transposition:** Extending the endofunctor to a *probabilistic* endofunctor incorporating target site preferences and epigenetic accessibility.
2. **TE ecosystem dynamics:** Modeling the competitive and cooperative interactions among different TE families as a categorical game.
3. **Horizontal transfer:** Formalizing the "import" of new TEs from other species as inter-genome functors.
4. **CRISPR as TE defense:** Connecting prokaryotic CRISPR-Cas systems to the TE framework as a "type-checking firewall."

13.5 Conclusion

The view of repetitive elements as self-modifying code resolves the "junk DNA" paradox: these elements are not genomic waste but the genome's metaprogramming layer. DNA transposons implement move semantics, retrotransposons implement copy semantics, satellite DNA provides structural alignment, ERVs are legacy imports, and domesticated elements demonstrate that even "parasitic" code can become essential infrastructure. The Gödelian self-prediction impossibility reveals a deep connection between genome evolution and the foundations of computation, suggesting that the creative unpredictability of evolution is not a bug but a fundamental feature of self-modifying systems.

References

[1] Lander, E.S., et al. Initial sequencing and analysis of the human genome. *Nature*, 409(6822):860–921, 2001.

[2] de Koning, A.P., Gu, W., Castoe, T.A., Batzer, M.A., and Pollock, D.D. Repetitive elements may comprise over two-thirds of the human genome. *PLoS Genetics*, 7(12):e1002384, 2011.

[3] Orgel, L.E. and Crick, F.H. Selfish DNA: the ultimate parasite. *Nature*, 284(5757):604–607, 1980.

[4] Doolittle, W.F. and Sapienza, C. Selfish genes, the phenotype paradigm and genome evolution. *Nature*, 284(5757):601–603, 1980.

[5] Feschotte, C. Transposable elements and the evolution of regulatory networks. *Nature Reviews Genetics*, 9(5):397–405, 2008.

[6] Kazazian, H.H. Mobile elements: drivers of genome evolution. *Science*, 303(5664):1626–1632, 2004.

[7] Agrawal, A., Eastman, Q.M., and Schatz, D.G. Transposition mediated by RAG1 and RAG2 and its implications for the evolution of the immune system. *Nature*, 394(6695):744–751, 1998.

[8] Hiom, K., Melek, M., and Gellert, M. DNA transposition by the RAG1 and RAG2 proteins: a possible source of oncogenic translocations. *Cell*, 94(4):463–470, 1998.

[9] SanMiguel, P., Gaut, B.S., Tikhonov, A., Nakajima, Y., and Bennetzen, J.L. The paleontology of intergene retrotransposons of maize. *Nature Genetics*, 20(1):43–45, 1998.

[10] Mi, S., Lee, X., Li, X., Veldman, G.M., Bhatt, R.S., et al. Syncytin is a captive retroviral envelope protein involved in human placental morphogenesis. *Nature*, 403(6771):785–789, 2000.

[11] Wadsworth, C. and Bhatt, P. Modular composition in biological systems: from molecular to organismal scales. *Journal of Theoretical Biology*, 261(3):401–412, 2009.

[12] Awodey, S. *Category Theory*. Oxford University Press, 2nd edition, 2010.

[13] Pierce, B.C. *Types and Programming Languages*. MIT Press, 2002.

[14] Mac Lane, S. *Categories for the Working Mathematician*. Springer, 2nd edition, 1998.

[15] Lawvere, F.W. Diagonal arguments and cartesian closed categories. *Lecture Notes in Mathematics*, 92:134–145, 1969.

[16] Thompson, P.J., Macfarlan, T.S., and Lorincz, M.C. Long terminal repeats: from parasitic elements to building blocks of the transcriptional regulatory repertoire. *Molecular Cell*, 62(5):766–776, 2016.

[17] Bourque, G., Burns, K.H., Gehring, M., et al. Ten things you should know about transposable elements. *Genome Biology*, 19:199, 2018.