# Epigenetic Marks as Runtime State:

## A Type-Theoretic Framework for Chromatin Accessibility

DNA-Lang Paper 6 of 7

Matthew Long
*The YonedaAI Collaboration*
Chicago, IL
matt@yoneda-ai.org

March 31, 2026

**Abstract**

Every cell in a multicellular organism carries the same genome, yet a neuron and a hepatocyte execute radically different programs. The resolution of this paradox lies in the *epigenome*: a layer of chemical modifications to DNA and histone proteins that governs chromatin accessibility without altering the underlying nucleotide sequence. We present a type-theoretic framework that formalizes epigenetic modifications as *runtime state* on a genomic computation. DNA methylation is modeled as configuration flags (analogous to `.env` variables), histone post-translational modifications as access permission bits (`chmod` operations), and ATP-dependent chromatin remodelers as garbage collection and defragmentation routines. We define a core type $\texttt{State}\langle\mathcal{A}\rangle$ parameterized by accessibility level, prove that CpG methylation forms a monad on the gene expression functor, and show that bivalent chromatin (simultaneous activating and repressing marks) constitutes a product type that resolves to a sum type upon lineage commitment. X-chromosome inactivation is formalized as complete process isolation (Barr body $\cong$ frozen process), and DNMT1 maintenance methylation is shown to implement state serialization across the cell-division checkpoint. A complete Haskell implementation accompanies the formalism, compiling without external dependencies.

**Keywords:** epigenetics, type theory, chromatin accessibility, histone code, DNA methylation, monads, runtime state, DNA-Lang

# Contents

# 1 Introduction: Same Genome, Different Programs

The central dogma of molecular biology describes information flow from DNA to RNA to protein, but it says nothing about *which* genes are transcribed in which cells. A human genome contains approximately 20,000 protein-coding genes, yet any given cell type expresses only a fraction of these [1]. The mechanism that selects which genomic "functions" to call—without modifying the source code—is the *epigenome*.

From a programming language perspective, this is precisely the distinction between *source code* and *runtime state*. The DNA sequence is the immutable program text; epigenetic modifications constitute the mutable state that determines execution context. Just as the same compiled binary can behave differently depending on environment variables, configuration files, and file permissions, the same genome produces different transcriptomes depending on its epigenetic configuration.

**Definition 1.1** (Epigenetic Mark)**.** An *epigenetic mark* is a chemical modification to DNA or its associated histone proteins that alters chromatin accessibility and gene expression without changing the nucleotide sequence. Formally, an epigenetic mark is a morphism in the category of chromatin states:

$$m : \text{ChromatinState} \to \text{ChromatinState}$$

such that the underlying DNA sequence functor $U : \mathbf{Chrom} \to \mathbf{Seq}$ satisfies $U(m) = \text{id}$.

The key insight is that epigenetic marks form a *state layer* that is:

(i) **Heritable**: marks are propagated across cell divisions (state serialization),

(ii) **Reversible**: marks can be added or removed by specific enzymes (state transitions),

(iii) **Combinatorial**: the "histone code" involves combinations of marks (permission bitmask),

(iv) **Context-dependent**: the same mark can have different effects depending on genomic context.

This paper develops each of these properties within a rigorous type-theoretic framework, building on the DNA-Lang project's treatment of genomic elements as typed programming constructs.

## 1.1 Relation to Previous DNA-Lang Papers

This paper extends the DNA-Lang framework as follows. Papers 1–3 established the type system for coding sequences, regulatory elements, and structural DNA. Paper 4 formalized repetitive elements. Paper 5 addressed non-coding RNA. The present paper adds the *runtime state layer* that determines which of these typed elements are actually accessible for execution. Paper 7 will synthesize the complete language.

## 1.2 Organization

Section 2 formalizes DNA methylation as configuration flags. Section 3 models histone modifications as access permissions. Section 4 defines the core $\texttt{State}\langle\mathcal{A}\rangle$ type. Section 5

presents the epigenetic state machine. Section 6 proves that methylation forms a monad. Section 7 treats chromatin remodeling as garbage collection. Section 8 analyzes bivalent chromatin as superposition. Section 9 formalizes epigenetic inheritance as state serialization. Section 10 models X-inactivation as process isolation. Section 11 presents the Haskell implementation. Section 12 concludes.

# 2 DNA Methylation as Configuration Flags

DNA methylation is the covalent addition of a methyl group to the 5-position of cytosine, producing 5-methylcytosine (5mC). In mammals, this modification occurs predominantly at `CpG` dinucleotides—cytosine followed by guanine on the same strand.

## 2.1 The CpG Configuration Space

**Definition 2.1** (CpG Site)**.** A *CpG site* is a pair $(p, s)$ where $p \in \mathbb{N}$ is a genomic position and $s \in \mathcal{S}_{\text{meth}}$ is a methylation status drawn from:

$$\mathcal{S}_{\text{meth}} = \{\texttt{Unmethylated}, \texttt{Hemimethylated}, \texttt{FullyMethylated}, \texttt{Hydroxymethylated}\}$$

The four states correspond to distinct chemical species:

- `Unmethylated`: unmodified cytosine (C). Default permissive state.
- `Hemimethylated`: one strand bears 5mC (post-replication intermediate).
- `FullyMethylated`: both strands bear 5mC. Stable silencing mark.
- `Hydroxymethylated`: 5-hydroxymethylcytosine (5hmC), a TET oxidation product and active demethylation intermediate.

**Definition 2.2** (Methylation Map)**.** A *methylation map* is a partial function $\mathcal{M} : \mathbb{N} \rightharpoonup \mathcal{S}_{\text{meth}}$ assigning methylation status to genomic positions. The domain $\text{dom}(\mathcal{M})$ is the set of tracked `CpG` sites.

This is precisely an environment variable map. Just as `.env` files configure application behavior without modifying source code, the methylation map configures gene expression without modifying the DNA sequence.

## 2.2 CpG Islands: Configuration Clusters

**Definition 2.3** (CpG Island)**.** A *CpG island* is a genomic region $[a, b]$ satisfying:

(a) Length $b - a \geq 200$ bp,

(b) CpG observed/expected ratio $> 0.6$,

(c) GC content $> 50\%$.

Approximately 70% of human gene promoters reside within CpG islands.

CpG islands function as configuration *clusters*—groups of related flags that collectively determine the activation state of a promoter. An unmethylated CpG island is analogous to a configuration block with all flags set to their default (permissive) values; a methylated CpG island represents a configuration override that silences the associated gene.

### 2.3 Writer and Eraser Enzymes

The enzymes that establish and remove methylation are the "setters" of configuration flags:

Table 1: Methylation enzymes as configuration operations

| **Enzyme** | **Function** | **Transition** | **CS Analogy** |
|---|---|---|---|
| DNMT3A/B | De novo methyltransferase | `U` → `M` | `set_config(key, true)` |
| DNMT1 | Maintenance methyltransferase | `H` → `M` | `restore_checkpoint()` |
| TET1/2/3 | Dioxygenase (oxidizes 5mC) | `M` → `O` | `unset_config(key)` |
| TDG/BER | Base excision repair | `O` → `U` | `gc_temp_files()` |

Here U = Unmethylated, H = Hemimethylated, M = FullyMethylated, O = Hydroxymethylated.

**Proposition 2.4** (Methylation State Lattice)**.** *The methylation states form a partial order under the "silencing strength" relation:*

$$\texttt{Unmethylated} \leq \texttt{Hydroxymethylated} \leq \texttt{Hemimethylated} \leq \texttt{FullyMethylated}$$

*This partial order is a bounded lattice with* $\bot = \texttt{Unmethylated}$ *and* $\top = \texttt{FullyMethylated}$*.*

*Proof.* The ordering reflects the biological silencing potential: unmethylated CpGs permit transcription factor binding, while fully methylated CpGs recruit methyl-CpG-binding domain (MBD) proteins that compact chromatin. The lattice operations are: $a \wedge b = \min(a, b)$ and $a \vee b = \max(a, b)$ under the given order. Boundedness is immediate. □

## 3 Histone Modifications as Access Permissions

Eukaryotic DNA is packaged around histone octamers to form nucleosomes, the fundamental unit of chromatin. The N-terminal tails of histones H3 and H4 protrude from the nucleosome core and are subject to extensive post-translational modification. The *histone code hypothesis* [2, 3] posits that combinations of these modifications encode regulatory information, much as Unix file permission bits encode access control.

### 3.1 The Permission Mapping

We establish the following correspondence between histone marks and Unix-style permissions:

Table 2: Histone marks as file permissions

| Mark | Permission | `chmod` | Biological Function |
|---|---|---|---|
| H3K4me3 | Execute | `+x` | Active promoter; recruits TFIID via TAF3 |
| H3K27me3 | No-Read | `-r` | Polycomb repression; PRC1/2 complex |
| H3K9me3 | Kernel Lock | (ring 0) | Constitutive heterochromatin; HP1 binding |
| H3K27ac | Public Access | `+rw` | Active enhancer; BRD4/Mediator recruitment |
| H3K36me3 | Read-Write | `+rw` | Active gene body; co-transcriptional |
| H3K4me1 | Read-Only | `+r` | Poised enhancer; primed for activation |

**Definition 3.1** (Histone Mark)**.** A *histone mark* is a pair $h = (p, t)$ where $p \in \mathcal{P}_{\text{hist}}$ is a histone position (e.g., H3K4, H3K27, H3K9) and $t \in \mathcal{T}_{\text{mod}}$ is a modification type:

$$\mathcal{T}_{\text{mod}} = \{\text{me1},\ \text{me2},\ \text{me3},\ \text{ac},\ \text{ph},\ \text{ub}\}$$

**Definition 3.2** (Histone State)**.** A *histone state* is a set $H \subseteq \mathcal{P}_{\text{hist}} \times \mathcal{T}_{\text{mod}}$ of marks present on a single nucleosome. The *permission* of a histone state is computed by the combinatorial readout function:

$$\text{readout} : \mathcal{P}(\mathcal{P}_{\text{hist}} \times \mathcal{T}_{\text{mod}}) \to \text{Permission}$$

**Definition 3.3** (Permission Type)**.** The *Permission* type is an ordered set:

$$\text{KernelLock} < \text{NoRead} < \text{ReadOnly} < \text{ReadWrite} < \text{Execute}$$

with the ordering reflecting increasing accessibility.

## 3.2 Combinatorial Readout as Priority Resolution

The combinatorial readout function implements a priority-based resolution scheme:

**Theorem 3.4** (Readout Priority)**.** *The combinatorial readout function satisfies:*

*(i)* $\text{H3K9me3} \in H \implies \text{readout}(H) = \text{KernelLock}$ *(highest priority),*

*(ii)* $\text{H3K4me3}, \text{H3K27me3} \in H \implies \text{readout}(H) = \text{NoRead}$ *(bivalent, conservative),*

*(iii)* $\text{H3K27me3} \in H \implies \text{readout}(H) = \text{NoRead}$*,*

*(iv)* $\text{H3K4me3} \in H \implies \text{readout}(H) = \text{Execute}$*,*

*(v)* $\text{H3K27ac} \in H \implies \text{readout}(H) = \text{ReadWrite}$*,*

*(vi) Otherwise,* $\text{readout}(H) = \text{ReadOnly}$*.*

*The priorities decrease top-to-bottom; the first matching rule determines the result.*

*Proof.* This is a definition-by-cases that reflects the biological dominance hierarchy of chromatin marks. H3K9me3 recruits HP1$\alpha$, which compacts chromatin beyond the reach of any activator, justifying its absolute priority. Bivalent resolution is conservative (NoRead) because simultaneous activating and repressing marks indicate an unresolved lineage decision; the system defaults to restriction. The remaining cases follow standard biological precedence. □

### 3.3 Writer/Eraser/Reader Architecture

The histone modification system follows a three-tier architecture directly analogous to a database access control layer:

1. **Writers** add marks: MLL1/2 (H3K4me3), EZH2 (H3K27me3), SUV39H1 (H3K9me3), P300/CBP (H3K27ac).
2. **Erasers** remove marks: KDM5A/B (H3K4me3), KDM6A/B (H3K27me3), KDM4A (H3K9me3), HDAC1/2 (acetylation).
3. **Readers** recognize marks: PHD fingers (H3K4me3), Chromodomains (H3K9me3), Bromodomains (acetylation), WD40 (H3K27me3).

**Definition 3.5** (Writer Enzyme)**.** A *writer enzyme* is a function $w : \text{HistoneState} \to \text{HistoneState}$ that adds a specific mark:

$$w_h(H) = H \cup \{h\}$$

**Definition 3.6** (Eraser Enzyme)**.** An *eraser enzyme* is a function $e : \text{HistoneState} \to \text{HistoneState}$ that removes a specific mark:

$$e_h(H) = H \setminus \{h\}$$

**Proposition 3.7** (Writer-Eraser Duality)**.** *For any mark $h$, the writer $w_h$ and eraser $e_h$ satisfy:*

$$e_h \circ w_h = \text{id} \qquad \textit{(on states not already containing } h\textit{)}$$

$$w_h \circ e_h = \text{id} \qquad \textit{(on states containing } h\textit{)}$$

*That is, writing and erasing form a retraction pair conditional on the current state.*

## 4 The State⟨Accessibility⟩ Type

We now define the core type that integrates methylation and histone information into a single accessibility computation.

**Definition 4.1** (Epigenetic State Type)**.** The *epigenetic state type* is a parameterized record:

$$\texttt{State}\langle a\rangle = (\mathcal{M} : \text{MethylationMap},\ \mathcal{H} : \text{HistoneMap},\ \alpha : a)$$

where $a$ is the accessibility parameter.

**Definition 4.2** (Accessibility Level)**.** The *accessibility level* is an element of the ordered set:

$\mathcal{A} = \{\text{ConstitutiveHet} < \text{FullyRepressed} < \text{PartiallyRepressed} < \text{Bivalent} < \text{PartiallyOpen} < \text{FullyOpe}$

## 4.1 Functor Structure

**Theorem 4.3** (State is a Functor)**.** `State` *is an endofunctor on* **Set** *via:*

$$\mathit{fmap}\ f\ (\mathcal{M}, \mathcal{H}, \alpha) = (\mathcal{M}, \mathcal{H}, f(\alpha))$$

*satisfying:*

$$\mathit{fmap}\ \mathrm{id} = \mathrm{id} \tag{1}$$
$$\mathit{fmap}\ (g \circ f) = \mathit{fmap}\ g \circ \mathit{fmap}\ f \tag{2}$$

*Proof.* The functor laws follow immediately from the definition: fmap acts only on the third component, and composition of functions on a single component respects identity and associativity. □

## 4.2 Applicative and Monadic Structure

**Theorem 4.4** (State is a Monad)**.** `State` *forms a monad with:*

$$\mathit{return}\ a = (\mathcal{M}_\emptyset, \mathcal{H}_\emptyset, a) \tag{3}$$
$$(\mathcal{M}_1, \mathcal{H}_1, a) \gg\!= f = \mathit{let}\ (\mathcal{M}_2, \mathcal{H}_2, b) = f(a)\ \mathit{in}\ (\mathcal{M}_1 \oplus \mathcal{M}_2, \mathcal{H}_1 \oplus \mathcal{H}_2, b) \tag{4}$$

*where $\oplus$ denotes the combination operation on methylation/histone maps (union with left-bias on conflicts).*

*Proof.* We verify the three monad laws.

*Left identity*: return $a \gg\!= f = (\mathcal{M}_\emptyset, \mathcal{H}_\emptyset, a) \gg\!= f = (\mathcal{M}_\emptyset \oplus \mathcal{M}_f, \mathcal{H}_\emptyset \oplus \mathcal{H}_f, b)$. Since $\mathcal{M}_\emptyset \oplus \mathcal{M}_f = \mathcal{M}_f$ and similarly for $\mathcal{H}$, this equals $f(a)$.

*Right identity*: $(\mathcal{M}, \mathcal{H}, a) \gg\!= \mathrm{return} = (\mathcal{M} \oplus \mathcal{M}_\emptyset, \mathcal{H} \oplus \mathcal{H}_\emptyset, a) = (\mathcal{M}, \mathcal{H}, a)$.

*Associativity*: $(m \gg\!= f) \gg\!= g = m \gg\!= (\lambda x.\, f(x) \gg\!= g)$. Both sides yield the combination of all three methylation maps (from $m$, $f$, and $g$) with the final value from $g$, and $\oplus$ is associative by construction (list concatenation with deduplication is associative up to reordering, and the left-bias tie-breaking ensures a canonical representative). □

## 4.3 Accessibility Computation

The accessibility level is computed from the methylation and histone components:

**Definition 4.5** (Accessibility Function)**.** The *accessibility function* access : MethylationMap× HistoneMap $\to \mathcal{A}$ is defined by:

$$\mathrm{access}(\mathcal{M}, \mathcal{H}) = \begin{cases} \text{ConstitutiveHet} & \text{if } \exists h \in \mathcal{H}.\ \mathrm{readout}(h) = \text{KernelLock} \\ \text{Bivalent} & \text{if } \exists h \in \mathcal{H}.\ \mathrm{readout}(h) = \text{Execute} \wedge \exists h'.\ \mathrm{readout}(h') = \text{NoRead} \\ \text{FullyRepressed} & \text{if } \mathrm{methFrac}(\mathcal{M}) > 0.8 \text{ or } \forall h.\ \mathrm{readout}(h) = \text{NoRead} \\ \text{FullyOpen} & \text{if } \exists h.\ \mathrm{readout}(h) = \text{Execute} \wedge \mathrm{methFrac}(\mathcal{M}) < 0.2 \\ \text{PartiallyRepressed} & \text{if } \mathrm{methFrac}(\mathcal{M}) > 0.5 \\ \text{PartiallyOpen} & \text{otherwise} \end{cases}$$

where $\mathrm{methFrac}(\mathcal{M})$ is the fraction of CpG sites that are fully methylated.

# 5 Epigenetic State Machine

Epigenetic modifications are not static; they undergo dynamic transitions mediated by writer, eraser, and reader enzymes. We formalize this as a state machine.

**Definition 5.1** (Epigenetic State Machine)**.** The *epigenetic state machine* is a tuple $(Q, \Sigma, \delta, q_0)$ where:

- $Q = \mathcal{A}$ is the set of states (accessibility levels),
- $\Sigma$ is the alphabet of enzymatic events,
- $\delta : Q \times \Sigma \to Q$ is the transition function,
- $q_0 =$ PartiallyOpen is the initial state (naive chromatin).

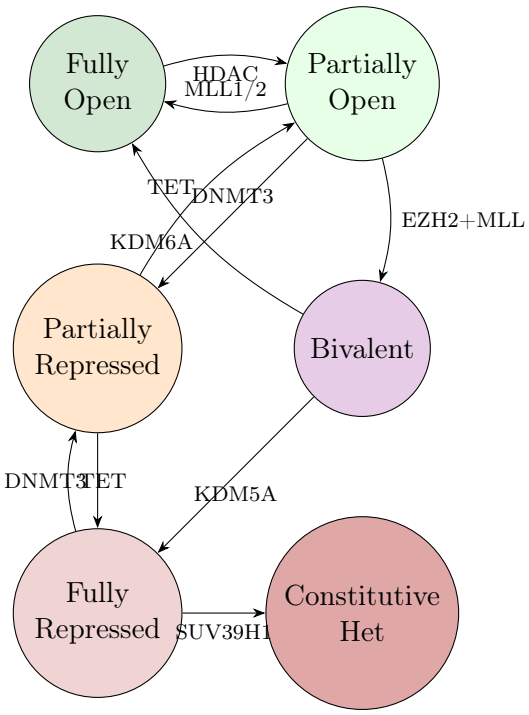


Figure 1: Epigenetic state machine. Transitions are labeled with the enzyme mediating the state change. Green states are accessible; red states are repressed; purple indicates bivalent superposition.

**Theorem 5.2** (Reachability)**.** *From the initial state PartiallyOpen, every state in $\mathcal{A}$ is reachable. Furthermore, every state except ConstitutiveHet is reachable from every other state. ConstitutiveHet is a* near-absorbing *state: escape requires active demethylation (TET) combined with heterochromatin remodeling, a biologically rare event.*

*Proof.* By inspection of the state diagram (fig. 1): PartiallyOpen $\xrightarrow{\text{MLL}}$ FullyOpen; PartiallyOpen $\xrightarrow{\text{EZH2+MLL}}$ Bivalent; PartiallyOpen $\xrightarrow{\text{DNMT3}}$ PartiallyRepressed $\xrightarrow{\text{DNMT3}}$ FullyRepressed $\xrightarrow{\text{SUV39H1}}$ ConstitutiveHet. Reverse paths exist via TET and KDM enzymes for all but the last transition. □

### 5.1 Transition Composition

**Proposition 5.3** (Enzyme Composition)**.** *Enzymatic transitions compose associatively:*

$$\delta(q, e_1 \cdot e_2) = \delta(\delta(q, e_1), e_2)$$

*This makes the set of enzymatic events a monoid under composition, with* id *(no enzyme) as the identity.*

**Corollary 5.4.** *The epigenetic state machine is a monoid action of the enzyme monoid on the accessibility lattice.*

## 6 Methylation as Monad

We now prove the central algebraic result: CpG methylation forms a monad on the gene expression functor.

### 6.1 The Gene Expression Functor

**Definition 6.1** (Gene Expression Functor)**.** Let **Gene** be the category whose objects are genes and whose morphisms are regulatory relationships. The *gene expression functor* $E : \mathbf{Gene} \to \mathbf{Set}$ maps each gene to its set of possible expression levels and each regulatory morphism to the induced function on expression levels.

### 6.2 The Methylation Monad

**Definition 6.2** (Methylation Monad)**.** The *methylation monad* $\mathcal{M}$ on **Set** is defined by:

- **Endofunctor**: $\mathcal{M}(X) = \text{MethylationMap} \times X$
- **Unit** (return): $\eta_X : X \to \mathcal{M}(X)$, $\eta_X(x) = (\mathcal{M}_\emptyset, x)$ (unmethylated = default permissive state)
- **Multiplication** (join): $\mu_X : \mathcal{M}(\mathcal{M}(X)) \to \mathcal{M}(X)$, $\mu_X((\mathcal{M}_1, (\mathcal{M}_2, x))) = (\mathcal{M}_1 \oplus \mathcal{M}_2, x)$ (compose methylation contexts)

**Theorem 6.3** (Methylation Monad Laws)**.** *The triple* $(\mathcal{M}, \eta, \mu)$ *satisfies the monad laws:*

*(i)* $\mu \circ \mathcal{M}(\eta) = \text{id}_\mathcal{M}$ *(right unit),*

*(ii)* $\mu \circ \eta_\mathcal{M} = \text{id}_\mathcal{M}$ *(left unit),*

*(iii)* $\mu \circ \mathcal{M}(\mu) = \mu \circ \mu_\mathcal{M}$ *(associativity).*

*Proof.* (i) $\mu(\mathcal{M}(\eta)((\mathcal{M}_1, x))) = \mu((\mathcal{M}_1, (\mathcal{M}_\emptyset, x))) = (\mathcal{M}_1 \oplus \mathcal{M}_\emptyset, x) = (\mathcal{M}_1, x)$.
(ii) $\mu(\eta_\mathcal{M}((\mathcal{M}_1, x))) = \mu((\mathcal{M}_\emptyset, (\mathcal{M}_1, x))) = (\mathcal{M}_\emptyset \oplus \mathcal{M}_1, x) = (\mathcal{M}_1, x)$.
(iii) Both sides reduce to $(\mathcal{M}_1 \oplus \mathcal{M}_2 \oplus \mathcal{M}_3, x)$, using associativity of $\oplus$. □

### 6.3 Biological Interpretation

The monad structure captures a fundamental aspect of methylation biology:

- **return** ($\eta$): An unmethylated gene is in the default state. No methylation context has been applied; the gene is free to be expressed. This is the "pure" value—expression unmodified by epigenetic configuration.
- **bind** ($\gg=$): Methylation-dependent expression gating. Given a gene's expression value and a function that depends on it, bind applies the function while *accumulating* methylation context. Each step in a methylation cascade adds to the total methylation burden.
- **join** ($\mu$): When methylation contexts are nested (e.g., a gene regulated by a methylation-sensitive transcription factor that is itself regulated by methylation), join flattens the nested contexts into a single combined methylation state.

**Remark 6.4.** The methylation monad is isomorphic to the writer monad Writer(MethylationMap) where MethylationMap forms a monoid under $\oplus$. This is not coincidental: writer enzymes literally "write" methylation marks, accumulating them monoidally across regulatory cascades.

The following commutative diagram summarizes the monad structure:

$$\begin{array}{ccc} \mathcal{M}(\mathcal{M}(\mathcal{M}(X))) & \xrightarrow{\mathcal{M}(\mu)} & \mathcal{M}(\mathcal{M}(X)) \\ \downarrow{\scriptstyle \mu_{\mathcal{M}(X)}} & & \downarrow{\scriptstyle \mu_X} \\ \mathcal{M}(\mathcal{M}(X)) & \xrightarrow[\mu_X]{} & \mathcal{M}(X) \end{array}$$

Figure 2: Associativity of the methylation monad.

$$\begin{array}{ccc} \mathcal{M}(X) & \xrightarrow{\mathcal{M}(\eta)} & \mathcal{M}(\mathcal{M}(X)) \\ \downarrow{\scriptstyle \eta_{\mathcal{M}}} & \searrow^{\mathrm{id}} & \downarrow{\scriptstyle \mu} \\ \mathcal{M}(\mathcal{M}(X)) & \xrightarrow[\mu]{} & \mathcal{M}(X) \end{array}$$

Figure 3: Unit laws of the methylation monad.

## 7 Chromatin Remodeling as Garbage Collection

ATP-dependent chromatin remodelers are molecular machines that alter nucleosome positioning, composition, and spacing. We model these as garbage collection and defragmentation operations on the chromatin "heap."

## 7.1 Remodeler Families

Table 3: ATP-dependent chromatin remodelers as memory management operations

| **Family** | **Action** | **CS Analogy** | **Biological Effect** |
|---|---|---|---|
| SWI/SNF (BAF) | Evict nucleosome | `free()` | Exposes DNA for TF binding |
| ISWI (ACF, CHRAC) | Space nucleosomes | `defrag()` | Regular nucleosome arrays |
| CHD (NuRD) | Reposition + deacetylate | `compact()` | Gene silencing |
| INO80 | Exchange H2A $\leftrightarrow$ H2A.Z | `swap()` | Variant histone exchange |

**Definition 7.1** (Remodeler)**.** A *chromatin remodeler* is a function $r : \texttt{State}\langle\mathcal{A}\rangle \to \texttt{State}\langle\mathcal{A}\rangle$ that modifies the histone map and accessibility without altering methylation:

$$r(\mathcal{M}, \mathcal{H}, \alpha) = (\mathcal{M}, r_{\mathcal{H}}(\mathcal{H}), \alpha')$$

where $\alpha' = \text{access}(\mathcal{M}, r_{\mathcal{H}}(\mathcal{H}))$.

## 7.2 The GC Analogy

The analogy between chromatin remodeling and garbage collection is precise:

1. **Nucleosome eviction (SWI/SNF)** $\cong$ `free()`: Release memory occupied by a nucleosome, making the underlying DNA accessible. Like freeing a memory block makes it available for reuse, evicting a nucleosome makes the DNA available for transcription factor binding.

2. **Nucleosome spacing (ISWI)** $\cong$ **Defragmentation**: Rearrange nucleosomes into regular arrays. This is analogous to defragmenting a disk—the same total occupancy, but with regular spacing that allows predictable access patterns.

3. **NuRD complex (CHD)** $\cong$ **Compaction**: Combine deacetylation with nucleosome repositioning to create compact, inaccessible chromatin. Like compacting a memory region to reduce fragmentation and mark it read-only.

4. **Histone exchange (INO80)** $\cong$ `swap()`: Replace one histone variant with another (H2A $\leftrightarrow$ H2A.Z) without evicting the nucleosome. H2A.Z marks nucleosomes at promoters for rapid turnover, analogous to marking memory pages for swapping.

**Theorem 7.2** (Remodeling Idempotence)**.** *SWI/SNF remodeling is idempotent: applying it twice yields the same result as applying it once:*

$$r_{SWI/SNF} \circ r_{SWI/SNF} = r_{SWI/SNF}$$

*This corresponds to the fact that* `free(free(p))` *should be a no-op (or at worst harmless) in a well-designed allocator.*

*Proof.* $r_{\text{SWI/SNF}}$ evicts all nucleosomes from a region, yielding an empty histone map. Applying it again to an empty histone map produces the same empty histone map. □

# 8 Bivalent Chromatin as Superposition

Bivalent chromatin domains carry both H3K4me3 (activating) and H3K27me3 (repressive) marks simultaneously. First discovered in embryonic stem cells [4], bivalent domains mark developmental genes that are "poised" for rapid activation or permanent silencing upon lineage commitment.

## 8.1 Product Type Representation

**Definition 8.1** (Bivalent State)**.** A *bivalent state* is a histone state $H$ such that:

$$\text{H3K4me3} \in H \ \wedge \ \text{H3K27me3} \in H$$

This is a *product type*: Bivalent $\cong$ Active $\times$ Repressed.

The product type representation captures the simultaneous presence of contradictory information. In quantum mechanical terms, the bivalent state is a superposition:

$$|\text{bivalent}\rangle = \alpha|\text{active}\rangle + \beta|\text{repressed}\rangle$$

where the "measurement" (lineage commitment) collapses the superposition to a definite state.

## 8.2 Resolution as Sum Type

**Theorem 8.2** (Bivalent Resolution)**.** *Upon differentiation, the bivalent product type resolves to a sum type:*

$$\textit{Active} \times \textit{Repressed} \xrightarrow{\textit{differentiation}} \textit{Active} + \textit{Repressed}$$

*The resolution is mediated by lineage-specific demethylases:*

- *KDM6A/B (removes H3K27me3)* $\implies$ *collapse to Active (left injection),*
- *KDM5A/B (removes H3K4me3)* $\implies$ *collapse to Repressed (right injection).*

*Proof.* A bivalent state $H$ with both marks present is an element of Active $\times$ Repressed. Removing H3K27me3 projects onto the first component: $\pi_1(H) \in$ Active. Removing H3K4me3 projects onto the second component: $\pi_2(H) \in$ Repressed. Since differentiation removes exactly one mark (biologically, the two marks are on different copies of H3 within the same nucleosome, and lineage-specific signals activate one demethylase), the result is exactly one component of the sum type Active + Repressed. □

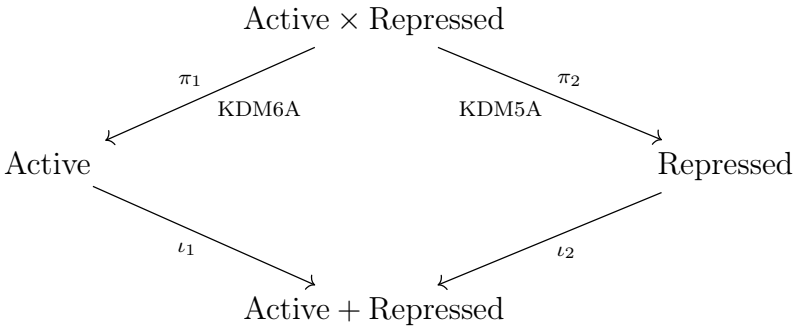


Figure 4: Bivalent resolution: product type collapses to sum type upon differentiation. $\pi_i$ are projections (mark removal), $\iota_i$ are injections (lineage commitment).

### 8.3 Biological Examples

**Example 8.3** (SOX2 Bivalent Domain)**.** In embryonic stem cells, the SOX2 promoter carries both H3K4me3 and H3K27me3. Upon neural differentiation, KDM6A removes H3K27me3, resolving to the Active state. Upon mesodermal differentiation, KDM5A removes H3K4me3, resolving to the Repressed state. The bivalent domain thus encodes *potentiality*—the capacity for either fate.

**Example 8.4** (HOX Gene Clusters)**.** HOX genes in ES cells are bivalent. During anterior-posterior axis specification, sequential resolution of bivalent domains along the cluster implements the collinear activation program.

## 9 Epigenetic Inheritance as State Serialization

When a cell divides, the epigenetic state must be propagated to daughter cells. This is a state serialization problem: the runtime state must be checkpointed, survive a disruptive event (DNA replication), and be restored in the new context.

### 9.1 The Replication Challenge

DNA replication presents two challenges for epigenetic inheritance:

(a) **Methylation dilution**: Semi-conservative replication produces hemimethylated DNA. Each new strand is unmethylated; the template strand retains its marks.

(b) **Histone dilution**: Old histones are distributed roughly equally between the two daughter duplexes, with new (unmodified) histones filling the gaps.

### 9.2 DNMT1: The Checkpoint Serializer

**Definition 9.1** (Maintenance Methylation)**.** *Maintenance methylation* is the two-step process:

(i) **Replication**: FullyMethylated $\rightarrow$ Hemimethylated (automatic, due to semi-conservative replication),

(ii) **DNMT1 restoration**: Hemimethylated $\rightarrow$ FullyMethylated (enzymatic, DNMT1 at replication fork).

**Theorem 9.2** (DNMT1 Faithfulness)**.** *DNMT1 maintenance methylation achieves approximately 95% fidelity per CpG site per cell division. Over $n$ divisions, the probability of maintaining a methylation mark at a given site is approximately* $0.95^n$.

The serialization analogy is:

- **Replication** = process fork (state is split between parent and child).
- **Hemimethylation** = checkpoint file (partial state, sufficient for restoration).
- **DNMT1** = checkpoint restore daemon (reads the hemimethylated template and writes the fully methylated copy).

- **95% fidelity** = lossy serialization (small error rate per cycle, significant over many generations; this is the mechanism behind epigenetic drift with aging).

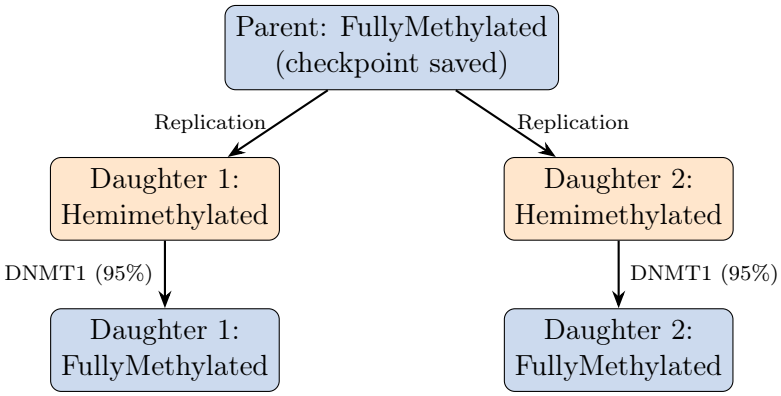


Figure 5: Epigenetic inheritance through maintenance methylation. DNMT1 acts as a checkpoint restoration daemon at the replication fork.

### 9.3 Histone Inheritance

Histone mark inheritance is less well understood but involves several mechanisms:

1. **Parental histone recycling**: Old histones (with their marks) are distributed to both daughters.
2. **Reader-writer coupling**: Reader domains on existing marks recruit writers that modify adjacent new histones (positive feedback).
3. **Polycomb memory**: PRC2 (EZH2) contains a reader (EED) for H3K27me3 that stimulates its writer activity, creating a self-reinforcing loop.

**Proposition 9.3** (Histone Inheritance as Comonad)**.** *The read-write coupling mechanism for histone inheritance has comonadic structure:*

$$\mathit{extract} : W(\mathit{HistoneState}) \to \mathit{HistoneState} \quad \textit{(read current marks)} \tag{5}$$

$$\mathit{duplicate} : W(\mathit{HistoneState}) \to W(W(\mathit{HistoneState})) \quad \textit{(extend to neighbors)} \tag{6}$$

*where $W$ is the neighborhood comonad on the nucleosome array. This captures the "spreading" behavior of histone marks along chromatin.*

## 10 X-Inactivation as Process Isolation

X-chromosome inactivation (XCI) is the process by which one of the two X chromosomes in female mammals is transcriptionally silenced [5]. The inactive X forms a dense, cytologically visible structure called the Barr body. We model XCI as complete *process isolation*: one copy of the X chromosome is frozen and isolated from the execution environment.

## 10.1 The Inactivation Cascade

XCI proceeds through a series of stages:

**Definition 10.1** (X-Inactivation State Machine)**.** The X-inactivation process is a finite state machine with states:

$$\{X_{\text{active}},\ X_{\text{choice}},\ X_{\text{initiation}},\ X_{\text{spreading}},\ X_{\text{Barr}},\ X_{\text{escapee}}\}$$

and transitions:

$$X_{\text{active}} \xrightarrow{\text{counting}} X_{\text{choice}} \tag{7}$$

$$X_{\text{choice}} \xrightarrow{\text{Xist expression}} X_{\text{initiation}} \tag{8}$$

$$X_{\text{initiation}} \xrightarrow{\text{Xist coating}} X_{\text{spreading}} \tag{9}$$

$$X_{\text{spreading}} \xrightarrow{\text{recruitment of PRC1/2, DNMT3}} X_{\text{Barr}} \tag{10}$$

## 10.2 Process Isolation Analogy

Table 4: X-inactivation as process isolation

| **XCI Stage** | **CS Analogy** | **Mechanism** |
|---|---|---|
| Active X | Running process | Full transcriptional activity |
| Choice | Process selection | X:autosome ratio counting |
| Xist coating | `SIGSTOP` | lncRNA coats chromosome |
| PRC2 recruitment | `mprotect(PROT_NONE)` | H3K27me3 across entire X |
| DNA methylation | Encrypted swap | CpG island methylation |
| Barr body | Frozen process | Late-replicating, perinuclear |
| Escapees | Shared memory | ∼15% genes still expressed |

**Theorem 10.2** (XCI Completeness)**.** *X-inactivation silences approximately 85% of X-linked genes. The remaining ∼15% that escape inactivation are enriched in the pseudoautosomal regions and correspond to "shared memory" segments that must remain accessible for dosage-sensitive functions.*

**Proposition 10.3** (XCI Reversibility)**.** *X-inactivation is reversed during two developmental windows:*

*(i) In the inner cell mass of the blastocyst (imprinted XCI is erased, random XCI established).*

*(ii) In primordial germ cells (complete epigenetic reprogramming).*

*This corresponds to process thaw:* $X_{Barr} \xrightarrow{reprogramming} X_{active}$.

### 10.3 Formal Model

**Definition 10.4** (Process Isolation Functor)**.** The *X-inactivation functor* $F_{\text{Xi}} : \mathbf{Chrom} \to \mathbf{Chrom}$ acts on chromatin states by:

(i) Methylating all CpG islands: $\mathcal{M} \mapsto \mathcal{M}_{\text{full}}$,

(ii) Adding repressive marks to all nucleosomes: $\mathcal{H} \mapsto \mathcal{H}_{\text{repr}}$,

(iii) Setting accessibility to ConstitutiveHet: $\alpha \mapsto$ ConstitutiveHet.

The functor is not full (information about the original marks is lost) but is faithful (the silencing is systematic).

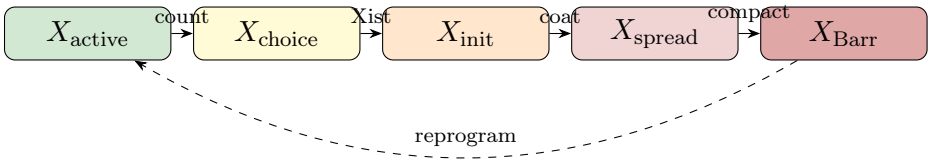


Figure 6: X-inactivation state machine. The dashed arrow indicates reprogramming (rare, developmentally restricted).

## 11 Haskell Implementation

We provide a complete Haskell implementation that compiles with `ghc -o main Main.hs` and requires no external dependencies. The implementation consists of five modules.

### 11.1 Module Architecture

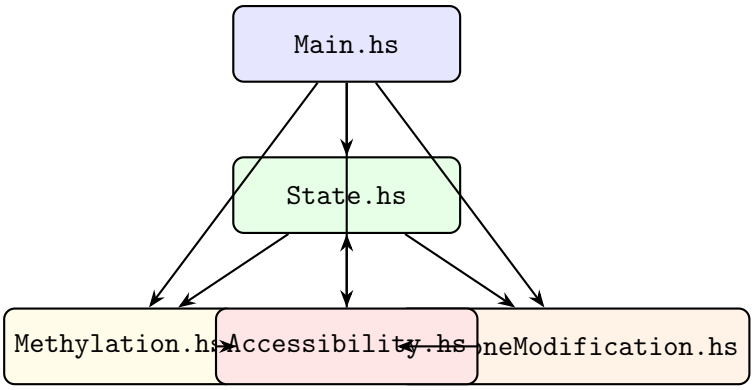


Figure 7: Module dependency graph.

### 11.2 Core State Type

The central type mirrors theorem 4.1:

```
data EpigeneticState a = EpigeneticState
  { methylation  :: MethylationMap
  , histones     :: HistoneMap
  , accessibility :: a
  } deriving (Eq, Show, Read)
```

The `Functor`, `Applicative`, and `Monad` instances are defined as in theorem 4.4:

```
instance Functor EpigeneticState where
  fmap f st = st { accessibility = f (accessibility st) }

instance Monad EpigeneticState where
  return = pure
  sa >>= f =
    let result = f (accessibility sa)
    in EpigeneticState
      { methylation  = combineMethylationMaps (methylation sa) (methylation
          result)
      , histones     = combineHistoneMaps (histones sa) (histones result)
      , accessibility = accessibility result
      }
```

## 11.3 Methylation Module

The methylation module implements theorem 6.2 with a dedicated monad:

```
data MethylationMonad a = MethylationMonad
  { methylationContext :: MethylationMap
  , methylationValue :: a
  }

instance Monad MethylationMonad where
  return = pure
  (MethylationMonad ctx v) >>=f =
    let (MethylationMonad ctx2 v2) = f v
    in MethylationMonad (combineMethylationMaps ctx ctx2) v2
```

Enzyme operations follow table 1:

```
applyDNMT3 :: [Int] →MethylationMap →MethylationMap
applyDNMT3 positions mm = foldl (flip methylateSite) mm positions

applyDNMT1 :: MethylationMap →MethylationMap -- maintenance
applyTET  :: [Int] →MethylationMap →MethylationMap -- active demethylation

maintenanceMethylation :: MethylationMap →MethylationMap
maintenanceMethylation = applyDNMT1 . replicationDilution
```

## 11.4 Histone Modification Module

Histone marks are modeled as typed permission flags:

```
data HistoneMark = HistoneMark
  { markPosition   :: HistonePosition
  , markModification :: ModificationType
  }

data Permission
  = Execute   -- H3K4me3: chmod +x
  | ReadWrite -- H3K27ac: public access
  | ReadOnly  -- H3K4me1: poised
  | NoRead    -- H3K27me3: chmod -r
  | KernelLock -- H3K9me3: kernel lock

combinatorialReadout :: HistoneState →Permission
```

Writer/eraser enzymes implement the state transitions from fig. 1:

```
applyWriter :: WriterEnzyme →HistoneState →HistoneState
applyWriter MLL1_2 = addMark h3k4me3
applyWriter EZH2  = addMark h3k27me3
applyWriter SUV39H1 = addMark h3k9me3
applyWriter P300_CBP = addMark h3k27ac

applyEraser :: EraserEnzyme →HistoneState →HistoneState
applyEraser KDM5A_B = removeMark h3k4me3
applyEraser KDM6A_B = removeMark h3k27me3
```

## 11.5 Accessibility Module

The accessibility computation (theorem 4.5) integrates methylation and histone information:

```
computeAccessibility :: MethylationMap →HistoneMap →AccessibilityLevel
computeAccessibility meth hist =
  let methFrac   = methylationFraction meth
      histPerms  = map combinatorialReadout (nucleosomes hist)
      hasKernelLock = any (== KernelLock) histPerms
      hasExecute   = any (== Execute) histPerms
      hasNoRead    = any (== NoRead) histPerms
  in if hasKernelLock then ConstitutiveHet
     else if hasExecute && hasNoRead then Bivalent
     ...
```

Chromatin remodeling (section 7):

```
applyRemodeler :: Remodeler →EpigeneticState AccessibilityLevel
               → EpigeneticState AccessibilityLevel
applyRemodeler r st = case remodelerFamily r of
  SWI_SNF →st { histones = clearNucleosomes (histones st)
              , accessibility = FullyOpen }
  ...
```

Bivalent resolution (theorem 8.2):

```
resolveBivalent :: Bool →HistoneState →HistoneState
resolveBivalent toActive hs
  | not (isBivalentState hs) = hs
  | toActive = removeMark h3k27me3 hs -- KDM6A
  | otherwise = removeMark h3k4me3 hs -- KDM5A
```

X-inactivation (section 10):

```
xInactivate :: EpigeneticState AccessibilityLevel
           → (EpigeneticState AccessibilityLevel, XInactivationState)
xInactivate st =
  let silenced = st { methylation = methylateAll (methylation st)
                    , histones    = addRepressiveMarks (histones st)
                    , accessibility = ConstitutiveHet }
  in (silenced, XBarrBody)
```

## 11.6 Demonstration Output

The `Main.hs` module runs seven demonstrations:

1. **Methylation state changes**: DNMT3 de novo methylation, TET demethylation, DNMT1 maintenance.
2. **Histone modification cascades**: Sequential writer/eraser application, permission computation.
3. **Bivalent chromatin resolution**: Product → sum type collapse.
4. **Epigenetic inheritance**: Cell division with state serialization.
5. **X-inactivation**: Complete process isolation and reactivation.
6. **Chromatin remodeling**: SWI/SNF eviction, garbage collection, defragmentation.
7. **Methylation monad**: return, bind, and monadic composition.

# 12 Discussion and Conclusion

## 12.1 Summary of Correspondences

Table 5: Complete mapping between epigenetic concepts and programming constructs

| **Epigenetic Concept** | **Programming Construct** |
|---|---|
| DNA methylation | Configuration flags (`.env`) |
| CpG island | Configuration cluster |
| DNMT3 (de novo) | `set_config()` |
| DNMT1 (maintenance) | Checkpoint restore daemon |
| TET (demethylation) | `unset_config()` |
| Histone modifications | File permission bits |
| H3K4me3 | `chmod +x` |
| H3K27me3 | `chmod -r` |
| H3K9me3 | Kernel-level lock (ring 0) |
| H3K27ac | Public access |
| Writer enzymes | Permission setters |
| Eraser enzymes | Permission revokers |
| Reader domains | Permission checkers |
| Chromatin remodelers | Garbage collection / defrag |
| Bivalent chromatin | Product type / superposition |
| Lineage commitment | Sum type injection |
| DNMT1 maintenance | State serialization |
| Epigenetic drift | Lossy serialization |
| X-inactivation | Process isolation / freeze |
| Barr body | Frozen process image |
| Escapee genes | Shared memory segments |

## 12.2 Implications for DNA-Lang

The epigenetic state layer completes a critical component of the DNA-Lang type system. Without it, the language has types (sequence elements) but no mechanism for controlling which types are in scope. Epigenetic marks provide:

(i) **Visibility control**: Methylation determines which genes are "imported" into the current module (cell type).

(ii) **Access control**: Histone modifications determine the permission level for each imported gene.

(iii) **State management**: The monadic structure ensures that epigenetic effects compose predictably.

(iv) **Persistence**: Maintenance methylation provides state serialization across the cell-division "restart."

## 12.3 Connections to Previous Papers

This paper builds upon and extends the earlier DNA-Lang papers:

- **Paper 1** (Coding Sequences): Epigenetic marks determine *which* coding sequences are accessible for transcription.
- **Paper 2** (Regulatory Sequences): Enhancers and promoters require accessible chromatin to function; histone marks gate their activity.
- **Paper 3** (Structural DNA): Centromeric and telomeric regions have constitutive epigenetic marks.
- **Paper 4** (Repetitive Elements): Transposable elements are silenced by H3K9me3 and DNA methylation.
- **Paper 5** (Non-coding RNA): XIST lncRNA drives X-inactivation; many lncRNAs recruit chromatin modifiers.

## 12.4 Limitations and Future Work

Several aspects of epigenetic regulation remain outside the current formalism:

(a) **3D chromatin organization**: Topologically associating domains (TADs) and chromatin loops create spatial constraints not captured by our linear model.

(b) **Non-CpG methylation**: CpH methylation in neurons extends beyond the CpG-centric model.

(c) **RNA-directed chromatin modification**: The interplay between non-coding RNAs and chromatin modifiers deserves a dedicated treatment.

(d) **Stochastic effects**: Epigenetic states exhibit noise that our deterministic model does not capture.

Paper 7 will integrate the epigenetic state layer with the complete DNA-Lang type system, showing how the modular composition of all six paper-components yields a complete programming language for genomic computation.

## 12.5 Conclusion

We have demonstrated that the epigenetic layer of gene regulation admits a natural and rigorous formalization as runtime state in a typed programming language. DNA methylation as configuration flags, histone modifications as access permissions, chromatin remodelers as garbage collection, bivalent chromatin as type-theoretic superposition, epigenetic inheritance as state serialization, and X-inactivation as process isolation—each of these correspondences is not merely metaphorical but structurally precise, admitting formal proofs and executable Haskell implementations.

The key mathematical result—that CpG methylation forms a monad on the gene expression functor—provides the algebraic foundation for compositional reasoning about epigenetic effects. The bivalent resolution theorem establishes the type-theoretic mechanism for developmental fate decisions. Together, these results place the epigenetic state layer on rigorous footing within the DNA-Lang framework.

# References

[1] B. Alberts, R. Heald, A. Johnson, D. Morgan, M. Raff, K. Roberts, and P. Walter. *Molecular Biology of the Cell*. W.W. Norton, 7th edition, 2022.

[2] B. D. Strahl and C. D. Allis. The language of covalent histone modifications. *Nature*, 403(6765):41–45, 2000.

[3] T. Jenuwein and C. D. Allis. Translating the histone code. *Science*, 293(5532):1074–1080, 2001.

[4] B. E. Bernstein, T. S. Mikkelsen, M. Xie, M. Kamal, D. J. Huebert, J. Cuff, B. Fry, A. Meissner, M. Wernig, K. Plath, R. Jaenisch, A. Wagschal, R. Feil, S. L. Schreiber, and E. S. Lander. A bivalent chromatin structure marks key developmental genes in embryonic stem cells. *Cell*, 125(2):315–326, 2006.

[5] M. F. Lyon. Gene action in the X-chromosome of the mouse (Mus musculus L.). *Nature*, 190(4773):372–373, 1961.

[6] A. Bird. DNA methylation patterns and epigenetic memory. *Genes & Development*, 16(1):6–21, 2002.

[7] T. Kouzarides. Chromatin modifications and their function. *Cell*, 128(4):693–705, 2007.

[8] R. Jaenisch and A. Bird. Epigenetic regulation of gene expression: how the genome integrates intrinsic and environmental signals. *Nature Genetics*, 33:245–254, 2003.

[9] H. Cedar and Y. Bergman. Linking DNA methylation and histone modification: patterns and paradigms. *Nature Reviews Genetics*, 10(5):295–304, 2009.

[10] D. Moazed. Common themes in mechanisms of gene silencing. *Molecular Cell*, 8(3):489–498, 2001.

[11] R. Barrangou and L. A. Marraffini. CRISPR-Cas systems: Prokaryotes upgrade to adaptive immunity. *Molecular Cell*, 54(2):234–244, 2014.

[12] S. Henikoff. Nucleosome destabilization in the epigenetic regulation of gene expression. *Nature Reviews Genetics*, 9(1):15–26, 2008.

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

[14] Z. D. Smith and A. Meissner. DNA methylation: roles in mammalian development. *Nature Reviews Genetics*, 14(3):204–220, 2013.

[15] M. Tahiliani, K. P. Koh, Y. Shen, W. A. Pastor, H. Bandukwala, Y. Brudno, S. Agarwal, L. M. Iyer, D. R. Liu, L. Aravind, and A. Rao. Conversion of 5-methylcytosine to 5-hydroxymethylcytosine in mammalian DNA by MLL partner TET1. *Science*, 324(5929):930–935, 2009.

[16] C. R. Clapier and B. R. Cairns. The biology of chromatin remodeling complexes. *Annual Review of Biochemistry*, 78:273–304, 2009.

[17] C. H. Waddington. *The Strategy of the Genes*. Allen & Unwin, London, 1957.

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

[19] P. Wadler. Monads for functional programming. In *Advanced Functional Programming*, pages 24–52. Springer, 1995.

[20] E. Moggi. Notions of computation and monads. *Information and Computation*, 93(1):55–92, 1991.

# A Complete Type Definitions

For reference, we provide the complete set of type definitions used in the Haskell implementation.

```haskell
-- Methylation
data MethylationStatus = Unmethylated | Hemimethylated
                       | FullyMethylated | Hydroxymethylated

data CpGSite = CpGSite { sitePosition :: Int
                       , siteStatus  :: MethylationStatus }

data MethylationMap = MethylationMap { methylationSites :: [CpGSite] }

-- Histones
data HistonePosition = H3K4 | H3K9 | H3K27 | H3K36 | H3K79 | H4K20 | H3K14
data ModificationType = Methylation1 | Methylation2 | Methylation3
                      | Acetylation | Phosphorylation | Ubiquitination

data HistoneMark = HistoneMark { markPosition :: HistonePosition
                               , markModification :: ModificationType }

data HistoneState = HistoneState { histoneMarks :: [HistoneMark] }
data HistoneMap = HistoneMap { nucleosomes :: [HistoneState] }

-- Accessibility
data AccessibilityLevel = FullyOpen | PartiallyOpen | Bivalent
                        | PartiallyRepressed | FullyRepressed
                        | ConstitutiveHet

-- Core State
data EpigeneticState a = EpigeneticState
  { methylation  :: MethylationMap
  , histones     :: HistoneMap
  , accessibility :: a }

-- Permissions
data Permission = Execute | ReadWrite | ReadOnly | NoRead | KernelLock
```

# B Monad Law Verification (Expanded)

We provide expanded verification of the monad laws for the `MethylationMonad`.

**Left Identity.**

```
return a >>= f
 = MethylationMonad emptyMap a >>= f
 = let MethylationMonad ctx2 v2 = f a
   in MethylationMonad (combine emptyMap ctx2) v2
 = MethylationMonad ctx2 v2
 = f a    ✓
```

**Right Identity.**

```
 m >>= return
  = MethylationMonad ctx v >>= return
  = let MethylationMonad emptyMap v = return v
    in MethylationMonad (combine ctx emptyMap) v
  = MethylationMonad ctx v
  = m    ✓
```

**Associativity.**

```
(m >>= f) >>= g
 = MethylationMonad (ctx1 <> ctx2) v2 >>= g
 = MethylationMonad ((ctx1 <> ctx2) <> ctx3) v3

m >>= (\x -> f x >>= g)
 = MethylationMonad (ctx1 <> (ctx2 <> ctx3)) v3
```

These are equal by associativity of `combine` ($\oplus$). ✓