A Functorial Formulation of Nuclear Fission:

Mass–Energy Equivalence, Binding Energy, and Chain-Reaction Dynamics in a Categorical Framework

Matthew Long
Independent Researcher
matthew@magneton.ai

December 2025

Abstract

We present a categorical reformulation of nuclear fission physics, treating nuclear states as objects in a symmetric monoidal category and fission processes as morphisms preserving essential structure. The binding energy function is elevated to a functor $\mathcal{B} : \mathbf{Nucl} \to \mathbf{Energy}$, and conservation laws (charge, baryon number, energy-momentum) emerge as natural transformations. Chain-reaction dynamics are modeled coalgebraically, with neutron multiplication captured by coalgebras over a branching functor. This formulation provides several advantages: (1) physical invariants become type-level constraints enforceable at compile time; (2) the compositional structure of nuclear processes is made explicit; (3) simulation and verification become amenable to proof-assistant technology. We provide complete Haskell implementations demonstrating these ideas, showing how categorical abstractions translate directly into executable, type-safe code. While the underlying physics is well-established, this structural reformulation offers new perspectives on nuclear processes and provides a template for categorical treatments of other physical theories.

Contents

1 Introduction

Nuclear fission, the splitting of heavy atomic nuclei into lighter fragments with the release of substantial energy, represents one of the most consequential physical processes harnessed by humanity. The fundamental physics—mass-energy equivalence, nuclear binding energies, and neutron-mediated chain reactions—has been understood since the pioneering work of Meitner, Frisch, Hahn, Strassmann, Fermi, and others in the late 1930s and early 1940s [Meitner and Frisch, 1939, Hahn and Strassmann, 1939, Fermi, 1946].

While the physics is classical and well-established, the *mathematical structure* underlying fission has not been systematically examined through the lens of modern category theory. This paper addresses that gap by providing a functorial formulation of nuclear fission that:

(i) Treats nuclear configurations as objects in a symmetric monoidal category **Nucl**

(ii) Models fission processes as morphisms preserving physical invariants

(iii) Elevates binding energy to a functor $\mathcal{B} : \mathbf{Nucl} \to \mathbf{Energy}$

(iv) Captures conservation laws as natural transformations

(v) Represents chain-reaction dynamics as coalgebras over branching functors

The benefits of this categorical reformulation are not merely aesthetic. By encoding physical structure in categorical terms, we gain:

- **Compositionality**: Complex processes decompose into simpler morphisms
- **Type Safety**: Physical invariants become compile-time constraints
- **Proof Assistance**: Categorical structures interface naturally with proof assistants
- **Generalization**: The framework extends to other nuclear and particle processes

We accompany the mathematical development with complete Haskell implementations, demonstrating that categorical abstractions translate directly into executable, type-safe code. This dual presentation—category theory alongside typed functional programming—exemplifies the "propositions as types" paradigm and shows how physical theories can be both formally specified and computationally realized.

1.1 Organization of This Paper

Section 2 reviews the classical mathematics of nuclear fission: mass-energy equivalence, the semi-empirical mass formula, energy partition, and chain-reaction dynamics. Section 3 develops the categorical framework, defining the category **Nucl** of nuclear states, the binding-energy functor, and conservation as natural transformations. Section 4 introduces the coalgebraic treatment of neutron branching and chain reactions. Section 5 presents complete Haskell implementations. Section 6 discusses applications and extensions. Section 8 concludes.

2 Classical Mathematics of Nuclear Fission

Before developing the categorical framework, we establish the classical mathematical foundations of nuclear fission. These results, while standard, are essential background and will be reformulated categorically in subsequent sections.

2.1 Mass-Energy Equivalence

The fundamental relationship underlying all nuclear energy processes is Einstein's mass-energy equivalence:

$$E = mc^2 \tag{1}$$

where E is energy, m is mass, and $c \approx 2.998 \times 10^8\,\mathrm{m/s}$ is the speed of light in vacuum.

For nuclear processes, we work with the *mass defect* Δm, the difference between the mass of reactants and products:

Definition 2.1 (Mass Defect)**.** For a nuclear reaction transforming initial state I to final state F, the mass defect is:

$$\Delta m = m_I - m_F \tag{2}$$

When $\Delta m > 0$, the reaction is exoergic, releasing energy:

$$Q = \Delta m \cdot c^2 \tag{3}$$

This Q-value represents the total energy available from the reaction.

Example 2.2 (Uranium-235 Fission)**.** Consider the canonical fission of $^{235}\mathrm{U}$ induced by thermal neutron capture:

$$^{235}\mathrm{U} + n \rightarrow {}^{236}\mathrm{U}^* \rightarrow {}^{141}\mathrm{Ba} + {}^{92}\mathrm{Kr} + 3n \tag{4}$$

The masses involved are:

$$m(^{235}\mathrm{U}) = 235.0439\,\mathrm{u} \tag{5}$$
$$m(n) = 1.0087\,\mathrm{u} \tag{6}$$
$$m(^{141}\mathrm{Ba}) = 140.9144\,\mathrm{u} \tag{7}$$
$$m(^{92}\mathrm{Kr}) = 91.9262\,\mathrm{u} \tag{8}$$

The mass defect is:

$$\Delta m = (235.0439 + 1.0087) - (140.9144 + 91.9262 + 3 \times 1.0087) \tag{9}$$
$$= 236.0526 - 235.8667 \tag{10}$$
$$= 0.1859\,\mathrm{u} \tag{11}$$

Using the conversion $1\,\mathrm{u} \cdot c^2 = 931.5\,\mathrm{MeV}$:

$$Q = 0.1859 \times 931.5 \approx 173\,\mathrm{MeV} \tag{12}$$

Including prompt gamma radiation and other contributions, the total energy release is approximately $200\,\mathrm{MeV}$ per fission.

2.2 Nuclear Binding Energy and the Semi-Empirical Mass Formula

The *binding energy* $B(A, Z)$ of a nucleus with mass number A and atomic number Z is the energy required to disassemble it into its constituent nucleons:

Definition 2.3 (Binding Energy)**.**

$$B(A, Z) = \left[Zm_p + (A - Z)m_n - m(A, Z)\right] c^2 \tag{13}$$

where m_p is the proton mass, m_n is the neutron mass, and $m(A, Z)$ is the nuclear mass.

The *semi-empirical mass formula* (SEMF), also known as the Bethe-Weizsäcker formula, provides an analytical approximation to binding energies:

Theorem 2.4 (Semi-Empirical Mass Formula)**.** *The binding energy of a nucleus* (A, Z) *is approximately:*

$$B(A,Z) = a_v A - a_s A^{2/3} - a_c \frac{Z^2}{A^{1/3}} - a_a \frac{(A-2Z)^2}{A} + \delta(A,Z) \tag{14}$$

where:

- $a_v \approx 15.8\,\text{MeV}$ *(volume term)*
- $a_s \approx 18.3\,\text{MeV}$ *(surface term)*
- $a_c \approx 0.714\,\text{MeV}$ *(Coulomb term)*
- $a_a \approx 23.2\,\text{MeV}$ *(asymmetry term)*
- $\delta(A,Z)$ *is the pairing term*

The pairing term $\delta(A,Z)$ accounts for the tendency of nucleons to pair:

$$\delta(A,Z) = \begin{cases} +\delta_0 A^{-1/2} & \text{even-even (Z even, N even)} \\ 0 & \text{odd A} \\ -\delta_0 A^{-1/2} & \text{odd-odd (Z odd, N odd)} \end{cases} \tag{15}$$

with $\delta_0 \approx 12\,\text{MeV}$.

2.2.1 Physical Interpretation of SEMF Terms

Each term in (14) has a clear physical origin:

1. **Volume Term** ($a_v A$): Reflects the saturation property of nuclear forces. Each nucleon interacts only with its nearest neighbors, so binding energy scales with volume ($\propto A$).
2. **Surface Term** ($-a_s A^{2/3}$): Surface nucleons have fewer neighbors than interior nucleons. Surface area scales as $A^{2/3}$, reducing binding.
3. **Coulomb Term** ($-a_c Z^2/A^{1/3}$): Protons repel electrostatically. For Z protons in a sphere of radius $R \propto A^{1/3}$, the Coulomb energy scales as $Z^2/R \propto Z^2/A^{1/3}$.
4. **Asymmetry Term** ($-a_a(A-2Z)^2/A$): The Pauli exclusion principle favors $N \approx Z$. Deviations incur an energy penalty proportional to $(N-Z)^2/A$.
5. **Pairing Term** (δ): Nucleons prefer to pair in spin-singlet states, stabilizing even-even nuclei.

2.2.2 Binding Energy Per Nucleon

The quantity of central importance for understanding fission (and fusion) is the binding energy per nucleon:

$$\frac{B}{A} = a_v - a_s A^{-1/3} - a_c \frac{Z^2}{A^{4/3}} - a_a \frac{(A-2Z)^2}{A^2} + \frac{\delta}{A} \tag{16}$$

This quantity exhibits a maximum near $A \approx 56$ (iron-nickel region) at approximately $8.8\,\text{MeV/nucleon}$. Heavy nuclei like uranium have $B/A \approx 7.6\,\text{MeV/nucleon}$, while fission fragments (medium-mass nuclei) have $B/A \approx 8.5\,\text{MeV/nucleon}$.

Proposition 2.5 (Fission Energy from Binding Energy Difference)**.** *The energy released in fission equals the increase in total binding energy:*

$$E_{fission} = \sum_{products} B_i - B_{parent} \tag{17}$$

Proof. By conservation of baryon number, $\sum_i A_i = A_{\text{parent}}$. The mass defect equals the binding energy difference divided by c^2, so:

$$E = \Delta m \cdot c^2 \tag{18}$$

$$= \left(m_{\text{parent}} - \sum_i m_i \right) c^2 \tag{19}$$

$$= \sum_i B_i - B_{\text{parent}} \tag{20}$$

where we used $m = Zm_p + Nm_n - B/c^2$. □

2.3 Energy Partition in Fission

The approximately 200 MeV released per ^{235}U fission is partitioned among several channels:

Component	**Energy (MeV)**	**Recoverable?**
Fragment kinetic energy	165	Yes
Prompt neutron kinetic energy	5	Yes
Prompt gamma radiation	7	Yes
Beta decay (delayed)	7	Yes
Gamma decay (delayed)	6	Yes
Antineutrinos	10	No
Total	∼200	∼190

Table 1: Energy partition in thermal neutron fission of ^{235}U.

The antineutrino energy escapes the reactor without interaction, leaving approximately 190 MeV as recoverable heat.

2.4 Chain Reaction Dynamics

Each fission event produces, on average, $\bar{\nu} \approx 2.4$ neutrons. These neutrons can induce further fissions, creating a chain reaction characterized by the *multiplication factor* k:

Definition 2.6 (Multiplication Factor)**.** The multiplication factor k is the ratio of neutrons in successive generations:

$$k = \frac{N_{n+1}}{N_n} \tag{21}$$

The chain reaction exhibits three regimes:

1. **Subcritical** ($k < 1$): Neutron population decays exponentially
2. **Critical** ($k = 1$): Neutron population is constant (steady-state reactor)
3. **Supercritical** ($k > 1$): Neutron population grows exponentially

2.4.1 Neutron Population Dynamics

In the discrete-generation model:

$$N_n = N_0 k^n \tag{22}$$

In continuous time, with generation time Λ:

$$N(t) = N_0 \exp\left(\frac{k-1}{\Lambda} t\right) \tag{23}$$

For thermal reactors, $\Lambda \approx 10^{-4}\,\text{s}$. Delayed neutrons (fraction $\beta \approx 0.0065$ for U-235) extend the effective generation time to $\Lambda_{\text{eff}} \approx 0.1\,\text{s}$, making reactor control feasible.

2.4.2 Reactor Power

For a reactor operating at fission rate R:

$$P = R \cdot E_{\text{fission}} \tag{24}$$

Example 2.7 (1 GW Reactor)**.** For a 1 GW thermal reactor:

$$R = \frac{10^9\,\text{J/s}}{200\,\text{MeV} \times 1.602 \times 10^{-13}\,\text{J/MeV}} \tag{25}$$

$$\approx 3.1 \times 10^{19}\,\text{fissions/s} \tag{26}$$

Fuel consumption: approximately 1 g of U-235 per megawatt-day.

3 Categorical Framework for Nuclear Fission

We now develop a categorical reformulation of the physics presented in Section 2. This is not merely a change of notation; the categorical perspective reveals compositional structure and enables type-safe implementations.

3.1 The Category of Nuclear States

Definition 3.1 (Category **Nucl**)**.** The category **Nucl** of nuclear states consists of:

Objects: Multisets of nuclides, where a nuclide is specified by (A, Z) pairs:

$$\text{ob}(\mathbf{Nucl}) = \left\{ \bigoplus_{i=1}^{n} (A_i, Z_i) : n \in \mathbb{N}, A_i, Z_i \in \mathbb{N}^+ \right\} \tag{27}$$

Morphisms: Physical processes transforming one nuclear configuration to another, subject to conservation laws:

$$\text{Hom}_{\mathbf{Nucl}}(X, Y) = \{\phi : X \to Y : \phi \text{ conserves } (A_{\text{tot}}, Z_{\text{tot}}, E_{\text{tot}}, \vec{p}_{\text{tot}})\} \tag{28}$$

Identity: The trivial process $\text{id}_X : X \to X$

Composition: Sequential application of processes: $(\psi \circ \phi)(x) = \psi(\phi(x))$

Proposition 3.2. Nucl *forms a category with the above definitions.*

Proof. We verify the category axioms:

1. **Identity**: $\text{id}_X \circ \phi = \phi = \phi \circ \text{id}_Y$ for $\phi : X \to Y$ (trivial).
2. **Associativity**: $(\chi \circ \psi) \circ \phi = \chi \circ (\psi \circ \phi)$ (composition of functions is associative).
3. **Closure**: Conservation laws are preserved under composition.

□

3.1.1 Monoidal Structure

Nucl carries additional structure making it a symmetric monoidal category:

Definition 3.3 (Monoidal Structure on **Nucl**)**.** $(\mathbf{Nucl}, \oplus, \emptyset)$ is a symmetric monoidal category where:

- $\oplus$ is multiset union (combining nuclear systems)
- $\emptyset$ is the empty multiset (vacuum)
- The braiding $\sigma_{X,Y} : X \oplus Y \xrightarrow{\sim} Y \oplus X$ is the canonical swap

The monoidal structure captures the physical idea that nuclear systems can be combined and that the order of combination is physically irrelevant.

3.2 The Binding Energy Functor

The binding energy function naturally extends to a functor:

Definition 3.4 (Binding Energy Functor)**.** The binding energy functor $\mathcal{B} : \mathbf{Nucl} \to \mathbf{Energy}$ is defined by:

On objects:

$$\mathcal{B}\left(\bigoplus_{i=1}^{n}(A_i, Z_i)\right) = \sum_{i=1}^{n} B(A_i, Z_i) \tag{29}$$

On morphisms: For $\phi : X \to Y$, define $\mathcal{B}(\phi) : \mathcal{B}(X) \to \mathcal{B}(Y)$ as the energy difference morphism:

$$\mathcal{B}(\phi) = \mathcal{B}(Y) - \mathcal{B}(X) \tag{30}$$

Theorem 3.5 (Functoriality of $\mathcal{B}$)**.** $\mathcal{B} : \mathbf{Nucl} \to \mathbf{Energy}$ *is a functor.*

Proof. We verify:

1. $\mathcal{B}(\mathrm{id}_X) = \mathcal{B}(X) - \mathcal{B}(X) = 0 = \mathrm{id}_{\mathcal{B}(X)}$
2. $\mathcal{B}(\psi \circ \phi) = \mathcal{B}(Z) - \mathcal{B}(X) = (\mathcal{B}(Z) - \mathcal{B}(Y)) + (\mathcal{B}(Y) - \mathcal{B}(X)) = \mathcal{B}(\psi) + \mathcal{B}(\phi) = \mathcal{B}(\psi) \circ \mathcal{B}(\phi)$

where composition in **Energy** is addition. □

Corollary 3.6 (Energy Release as Functor Image)**.** *The energy released in a fission process* $\phi : U \to F_1 \oplus F_2 \oplus \cdots$ *is:*

$$E_{release} = \mathcal{B}(\phi) = \mathcal{B}(F_1) + \mathcal{B}(F_2) + \cdots - \mathcal{B}(U) \tag{31}$$

3.3 Conservation Laws as Natural Transformations

Conservation laws in physics assert that certain quantities remain unchanged under physical processes. In categorical terms, these become natural transformations.

Definition 3.7 (Charge Functor)**.** The charge functor $Q : \mathbf{Nucl} \to \mathbb{Z}$ maps nuclear configurations to their total charge:

$$Q\left(\bigoplus_{i}(A_i, Z_i)\right) = \sum_{i} Z_i \tag{32}$$

Definition 3.8 (Baryon Number Functor)**.** The baryon number functor $A : \mathbf{Nucl} \to \mathbb{N}$ maps nuclear configurations to their total baryon number:

$$A\left(\bigoplus_{i}(A_i, Z_i)\right) = \sum_{i} A_i \tag{33}$$

Theorem 3.9 (Conservation as Natural Transformation)**.** *For any conserved quantity C, there is a natural transformation $\eta : C \Rightarrow C \circ F$ where F is the forgetful functor from* **Nucl** *to its underlying set.*

More precisely, conservation laws constrain which morphisms exist in **Nucl**:

Proposition 3.10. *A morphism $\phi : X \to Y$ exists in* **Nucl** *only if:*

$$Q(X) = Q(Y) \quad \textit{(charge conservation)} \tag{34}$$

$$A(X) = A(Y) \quad \textit{(baryon conservation)} \tag{35}$$

This categorical formulation makes conservation laws *structural*: illegal transitions simply have no corresponding morphism.

3.4 Fission as a Morphism

We can now precisely define fission in categorical terms:

Definition 3.11 (Fission Morphism)**.** A *fission morphism* is a morphism $\phi \in \mathrm{Hom}_{\mathbf{Nucl}}(X, Y)$ where:

1. $X = (A, Z)$ is a single heavy nucleus
2. $Y = \bigoplus_{i=1}^{n}(A_i, Z_i) \oplus k \cdot (1, 0)$ consists of fragments plus k free neutrons
3. $n \geq 2$ (at least two fragments)
4. $\mathcal{B}(\phi) > 0$ (energy-releasing)

Example 3.12 (U-235 Fission Morphism)**.** The fission of Example 2.2 corresponds to:

$$\phi : (236, 92) \to (141, 56) \oplus (92, 36) \oplus 3 \cdot (1, 0) \tag{36}$$

with $\mathcal{B}(\phi) \approx 173\,\mathrm{MeV}$.

3.5 The Fission Diagram

The fission process can be represented as a commutative diagram:

$$\begin{array}{ccccc} (A,Z) \oplus (1,0) & \xrightarrow{\text{capture}} & (A+1,Z)^* & \xrightarrow{\phi} & \bigoplus_i (A_i, Z_i) \oplus k(1,0) \\ \downarrow{\scriptstyle Q} & & \downarrow{\scriptstyle Q} & & \downarrow{\scriptstyle Q} \\ Z & = & Z & = & Z \end{array} \tag{37}$$

This diagram commutes precisely because charge is conserved at each step.

4 Coalgebraic Treatment of Chain Reactions

The chain reaction dynamics of Section 2.4 have a natural coalgebraic formulation. Coalgebras, the categorical dual of algebras, are particularly suited to modeling systems with branching or nondeterministic evolution.

4.1 The Branching Functor

Definition 4.1 (Branching Functor)**.** The branching functor $\mathcal{F} : \mathbf{Set} \to \mathbf{Set}$ is defined by:

$$\mathcal{F}(X) = \mathcal{P}_{\text{fin}}(X) \times \mathbb{R}_{\geq 0} \tag{38}$$

where $\mathcal{P}_{\text{fin}}$ denotes finite powersets and $\mathbb{R}_{\geq 0}$ tracks energy release.

Definition 4.2 (Fission Coalgebra)**.** A *fission coalgebra* is a pair (S, γ) where:

- S is a set of nuclear states
- $\gamma : S \to \mathcal{F}(S)$ is the transition function

For a fissile nucleus, γ maps it to the set of possible fission outcomes (with probabilities encoded as weights) and the energy released.

4.2 Neutron Multiplication as Coalgebra Morphism

The multiplication factor k emerges from the coalgebraic structure:

Definition 4.3 (Multiplication Coalgebra)**.** Define the multiplication coalgebra (N, μ) where:

- $N \cong \mathbb{N}$ represents neutron populations
- $\mu(n) = (kn, E_n)$ where k is the multiplication factor and E_n is the energy released

Proposition 4.4. *The iteration of μ recovers the discrete population dynamics:*

$$\mu^{(m)}(n_0) = (k^m n_0, E_{total}) \tag{39}$$

4.3 Final Coalgebra and Long-Term Behavior

The final coalgebra for $\mathcal{F}$ captures the "maximally informative" branching structure:

Theorem 4.5. *The final $\mathcal{F}$-coalgebra exists and consists of all possible infinite branching trees.*

For chain reactions:

- Subcritical ($k < 1$): Unique morphism to the final coalgebra factors through finite trees
- Critical ($k = 1$): Stationary behavior, morphism captures equilibrium
- Supercritical ($k > 1$): Unbounded trees, morphism captures exponential growth

4.4 Delayed Neutrons and Extended Coalgebras

Delayed neutrons require an extended coalgebraic treatment:

Definition 4.6 (Time-Extended Branching Functor)**.**

$$\mathcal{F}_t(X) = \mathcal{P}_{\text{fin}}(X \times \mathbb{R}_{>0}) \times \mathbb{R}_{\geq 0} \tag{40}$$

where the $\mathbb{R}_{>0}$ component tracks delay times.

This extension naturally incorporates the delayed neutron fraction β and the effective generation time Λ_{eff}.

5 Haskell Implementation

We now present complete Haskell implementations of the categorical structures developed above. The code demonstrates how categorical abstractions translate into executable, type-safe programs.

5.1 Basic Nuclear Data Types

```haskell
module FunctorialFission.Core where

-- | A nuclide is specified by mass number A and atomic number Z
data Nuclide = Nuclide
  { massNumber   :: !Int  -- ^ A (protons + neutrons)
  , atomicNumber :: !Int  -- ^ Z (protons)
  } deriving (Eq, Ord, Show)

-- | Neutron number
neutronNumber :: Nuclide -> Int
neutronNumber n = massNumber n - atomicNumber n

-- | Type alias for clarity
type MeV = Double

-- | A nuclear configuration is a multiset of nuclides
newtype NuclearConfig = NuclearConfig
  { nuclides :: [Nuclide]
  } deriving (Eq, Show)

-- | The monoidal unit (vacuum)
emptyConfig :: NuclearConfig
emptyConfig = NuclearConfig []

-- | Monoidal product (combining configurations)
combineConfigs :: NuclearConfig -> NuclearConfig -> NuclearConfig
combineConfigs (NuclearConfig xs) (NuclearConfig ys) =
  NuclearConfig (xs ++ ys)
```

Listing 1: Core nuclear data types

5.2 Semi-Empirical Mass Formula

```haskell
module FunctorialFission.SEMF where

import FunctorialFission.Core

-- | SEMF coefficients (in MeV)
data SEMFCoefficients = SEMFCoefficients
  { aVolume    :: !MeV  -- ^ Volume coefficient
  , aSurface   :: !MeV  -- ^ Surface coefficient
  , aCoulomb   :: !MeV  -- ^ Coulomb coefficient
  , aAsymmetry :: !MeV  -- ^ Asymmetry coefficient
  , aPairing   :: !MeV  -- ^ Pairing coefficient
  } deriving (Show)

-- | Standard SEMF coefficients
standardCoefficients :: SEMFCoefficients
standardCoefficients = SEMFCoefficients
```

```
  { aVolume    = 15.8
  , aSurface   = 18.3
  , aCoulomb   = 0.714
  , aAsymmetry = 23.2
  , aPairing   = 12.0
  }

-- | Pairing term classification
data Pairing = EvenEven | OddA | OddOdd
  deriving (Eq, Show)

-- | Determine pairing type
pairingType :: Nuclide -> Pairing
pairingType n
  | even z && even (a - z) = EvenEven
  | odd a                  = OddA
  | otherwise              = OddOdd
  where
    a = massNumber n
    z = atomicNumber n

-- | Calculate pairing term
pairingTerm :: SEMFCoefficients -> Nuclide -> MeV
pairingTerm coef n = case pairingType n of
  EvenEven -> aPairing coef / sqrt (fromIntegral a)
  OddA     -> 0
  OddOdd   -> -aPairing coef / sqrt (fromIntegral a)
  where a = massNumber n

-- | Calculate binding energy using SEMF
bindingEnergy :: SEMFCoefficients -> Nuclide -> MeV
bindingEnergy coef n = volume - surface - coulomb - asymmetry + pairing
  where
    a = fromIntegral (massNumber n)
    z = fromIntegral (atomicNumber n)

    volume    = aVolume coef * a
    surface   = aSurface coef * a ** (2/3)
    coulomb   = aCoulomb coef * z * z / (a ** (1/3))
    asymmetry = aAsymmetry coef * (a - 2*z)^2 / a
    pairing   = pairingTerm coef n

-- | Binding energy per nucleon
bindingPerNucleon :: SEMFCoefficients -> Nuclide -> MeV
bindingPerNucleon coef n =
  bindingEnergy coef n / fromIntegral (massNumber n)
```

Listing 2: SEMF implementation

5.3 The Binding Energy Functor

```
module FunctorialFission.Functor where

import FunctorialFission.Core
import FunctorialFission.SEMF

-- | The Binding Energy functor on objects
-- Maps a nuclear configuration to its total binding energy
```

```haskell
bindingFunctorObj :: SEMFCoefficients -> NuclearConfig -> MeV
bindingFunctorObj coef (NuclearConfig ns) =
  sum $ map (bindingEnergy coef) ns

-- | A fission morphism from source to target configuration
data FissionMorphism = FissionMorphism
  { source :: NuclearConfig
  , target :: NuclearConfig
  } deriving (Show)

-- | The Binding Energy functor on morphisms
-- Maps a fission morphism to the energy released
bindingFunctorMor :: SEMFCoefficients -> FissionMorphism -> MeV
bindingFunctorMor coef (FissionMorphism src tgt) =
  bindingFunctorObj coef tgt - bindingFunctorObj coef src

-- | Verify conservation laws for a morphism
data ConservationCheck = ConservationCheck
  { chargeConserved  :: Bool
  , baryonConserved  :: Bool
  } deriving (Show)

totalCharge :: NuclearConfig -> Int
totalCharge (NuclearConfig ns) = sum $ map atomicNumber ns

totalBaryon :: NuclearConfig -> Int
totalBaryon (NuclearConfig ns) = sum $ map massNumber ns

checkConservation :: FissionMorphism -> ConservationCheck
checkConservation (FissionMorphism src tgt) = ConservationCheck
  { chargeConserved = totalCharge src == totalCharge tgt
  , baryonConserved = totalBaryon src == totalBaryon tgt
  }

-- | Validate that a morphism is physically allowed
isValidMorphism :: FissionMorphism -> Bool
isValidMorphism m =
  chargeConserved check && baryonConserved check
  where check = checkConservation m
```

Listing 3: Binding energy functor implementation

5.4 Type-Safe Fission Processes

```haskell
{-# LANGUAGE GADTs #-}
{-# LANGUAGE DataKinds #-}
{-# LANGUAGE KindSignatures #-}
{-# LANGUAGE TypeFamilies #-}

module FunctorialFission.TypeSafe where

import GHC.TypeLits
import Data.Kind (Type)

-- | Type-level nuclide representation
data TNuclide (a :: Nat) (z :: Nat) = TNuclide

-- | Type-level nuclear configuration
```

```haskell
data TConfig :: [Type] -> Type where
  TEmpty :: TConfig '[]
  TCons  :: TNuclide a z -> TConfig rest -> TConfig (TNuclide a z ': rest)

-- | Type family for total charge
type family TotalZ (config :: [Type]) :: Nat where
  TotalZ '[] = 0
  TotalZ (TNuclide a z ': rest) = z + TotalZ rest

-- | Type family for total baryon number
type family TotalA (config :: [Type]) :: Nat where
  TotalA '[] = 0
  TotalA (TNuclide a z ': rest) = a + TotalA rest

-- | A type-safe fission morphism that enforces conservation at compile time
data SafeFission src tgt where
  SafeFission :: (TotalZ src ~ TotalZ tgt, TotalA src ~ TotalA tgt)
              => TConfig src -> TConfig tgt -> SafeFission src tgt

-- | Example: U-236 -> Ba-141 + Kr-92 + 3n
-- This will only compile if conservation laws are satisfied!
exampleFission :: SafeFission
  '[TNuclide 236 92]
  '[TNuclide 141 56, TNuclide 92 36, TNuclide 1 0, TNuclide 1 0, TNuclide 1 0]
exampleFission = SafeFission
  (TCons TNuclide TEmpty)
  (TCons TNuclide (TCons TNuclide (TCons TNuclide (TCons TNuclide (TCons TNuclide TEmpty)))))
```

Listing 4: Type-safe fission with GADTs

5.5 Coalgebraic Chain Reactions

```haskell
module FunctorialFission.Coalgebra where

import FunctorialFission.Core

-- | The branching functor F(X) = (P_fin(X), R+)
-- Represented as a list of outcomes with associated energy
data BranchF a = BranchF
  { outcomes     :: [a]       -- ^ Finite set of outcomes
  , energyRelease :: MeV      -- ^ Energy released
  } deriving (Show, Functor)

-- | A coalgebra is a type with a structure map
class Coalgebra f a where
  coalg :: a -> f a

-- | Neutron population state
data NeutronState = NeutronState
  { population :: !Double  -- ^ Number of neutrons
  , generation :: !Int     -- ^ Generation number
  } deriving (Show)

-- | Multiplication coalgebra instance
```

```haskell
instance Coalgebra BranchF NeutronState where
  coalg (NeutronState pop gen) = BranchF
    { outcomes = [NeutronState (k * pop) (gen + 1)]
    , energyRelease = pop * energyPerFission
    }
    where
      k = 1.0  -- Critical condition
      energyPerFission = 200.0  -- MeV

-- | Iterate the coalgebra n times
iterateCoalg :: (Coalgebra BranchF a) => Int -> a -> [BranchF a]
iterateCoalg 0 _ = []
iterateCoalg n x = let fx = coalg x
                   in fx : concatMap (iterateCoalg (n-1)) (outcomes fx)

-- | Total energy after n generations
totalEnergy :: Int -> NeutronState -> MeV
totalEnergy n initial = sum $ map energyRelease $ iterateCoalg n
   initial

-- | Delayed neutron extension
data DelayedNeutron = DelayedNeutron
  { delayTime :: !Double  -- ^ Delay time in seconds
  , fraction  :: !Double  -- ^ Fraction of total neutrons
  } deriving (Show)

-- | Extended coalgebra with delayed neutrons
data ExtendedBranch a = ExtendedBranch
  { promptOutcomes  :: [(a, Double)]  -- ^ (outcome, probability)
  , delayedOutcomes :: [(a, DelayedNeutron)]
  , totalEnergy     :: MeV
  } deriving (Show)
```

Listing 5: Coalgebraic chain reaction model

5.6 Reactor Simulation

```haskell
module FunctorialFission.Reactor where

import FunctorialFission.Core
import FunctorialFission.SEMF
import FunctorialFission.Coalgebra

-- | Reactor parameters
data ReactorParams = ReactorParams
  { thermalPower   :: !Double  -- ^ Watts
  , efficiency     :: !Double  -- ^ Thermal to electric
  , fuelMass       :: !Double  -- ^ kg of U-235
  , kEffective     :: !Double  -- ^ Effective multiplication factor
  } deriving (Show)

-- | Reactor state
data ReactorState = ReactorState
  { neutronPop    :: !Double  -- ^ Current neutron population
  , time          :: !Double  -- ^ Simulation time (s)
  , energyProduced :: !Double -- ^ Total energy produced (J)
  , fuelRemaining :: !Double  -- ^ Remaining fuel (kg)
  } deriving (Show)
```

```haskell

-- | Physical constants
mevToJoules :: Double
mevToJoules = 1.602e-13

avogadro :: Double
avogadro = 6.022e23

u235MolarMass :: Double
u235MolarMass = 235.0  -- g/mol

energyPerFissionJ :: Double
energyPerFissionJ = 200.0 * mevToJoules

-- | Fissions per second for given power
fissionRate :: Double -> Double
fissionRate power = power / energyPerFissionJ

-- | Fuel consumption rate (kg/s)
fuelConsumptionRate :: Double -> Double
fuelConsumptionRate power =
  (fissionRate power * u235MolarMass) / (avogadro * 1000)

-- | Simulate reactor for time dt
stepReactor :: Double -> ReactorParams -> ReactorState -> ReactorState
stepReactor dt params state = ReactorState
  { neutronPop = newPop
  , time = time state + dt
  , energyProduced = energyProduced state + power * dt
  , fuelRemaining = fuelRemaining state - consumption * dt
  }
  where
    k = kEffective params
    lambda = 0.0001  -- Generation time (s)
    newPop = neutronPop state * exp ((k - 1) * dt / lambda)
    power = thermalPower params
    consumption = fuelConsumptionRate power

-- | Run simulation for given duration
runSimulation :: Double -> Double -> ReactorParams -> ReactorState -> [
   ReactorState]
runSimulation duration dt params initial
  | time initial >= duration = [initial]
  | otherwise = initial : runSimulation duration dt params (stepReactor
    dt params initial)
```

Listing 6: Simple reactor simulation

5.7 Complete Example

```haskell
module FunctorialFission.Example where

import FunctorialFission.Core
import FunctorialFission.SEMF
import FunctorialFission.Functor

-- | Define key nuclides
uranium236 :: Nuclide
```

```haskell
uranium236 = Nuclide 236 92

barium141 :: Nuclide
barium141 = Nuclide 141 56

krypton92 :: Nuclide
krypton92 = Nuclide 92 36

neutron :: Nuclide
neutron = Nuclide 1 0

-- | Define configurations
initialConfig :: NuclearConfig
initialConfig = NuclearConfig [uranium236]

finalConfig :: NuclearConfig
finalConfig = NuclearConfig [barium141, krypton92, neutron, neutron,
    neutron]

-- | Define the fission morphism
u236Fission :: FissionMorphism
u236Fission = FissionMorphism initialConfig finalConfig

-- | Calculate energy release
main :: IO ()
main = do
  let coef = standardCoefficients

  putStrLn "=== Functorial Fission Analysis ==="
  putStrLn ""

  -- Binding energies
  putStrLn "Binding Energies (SEMF):"
  putStrLn $ "  U-236: " ++ show (bindingEnergy coef uranium236) ++ "
    MeV"
  putStrLn $ "  Ba-141: " ++ show (bindingEnergy coef barium141) ++ "
    MeV"
  putStrLn $ "  Kr-92: " ++ show (bindingEnergy coef krypton92) ++ "
    MeV"
  putStrLn ""

  -- Functor action
  putStrLn "Functor Action:"
  putStrLn $ "  B(initial): " ++ show (bindingFunctorObj coef
    initialConfig) ++ " MeV"
  putStrLn $ "  B(final): " ++ show (bindingFunctorObj coef finalConfig
    ) ++ " MeV"
  putStrLn $ "  B(phi) = Energy Release: " ++ show (bindingFunctorMor
    coef u236Fission) ++ " MeV"
  putStrLn ""

  -- Conservation check
  putStrLn "Conservation Laws:"
  let check = checkConservation u236Fission
  putStrLn $ "  Charge conserved: " ++ show (chargeConserved check)
  putStrLn $ "  Baryon conserved: " ++ show (baryonConserved check)
  putStrLn $ "  Valid morphism: " ++ show (isValidMorphism u236Fission)
```

Listing 7: Complete worked example

6 Applications and Extensions

The categorical framework developed here extends naturally to several related areas.

6.1 Fusion as the Dual Process

Nuclear fusion, the combining of light nuclei, is categorically dual to fission:

Definition 6.1 (Fusion Morphism)**.** A *fusion morphism* is $\psi : X \oplus Y \to Z$ where:

- X, Y are light nuclei
- Z is a heavier product
- $\mathcal{B}(\psi) > 0$

The entire framework translates: the binding functor still maps fusion morphisms to their energy release, conservation laws still constrain which morphisms exist, and chain reactions in fusion (as in hydrogen bombs) have coalgebraic formulations.

6.2 Nuclear Reaction Networks

Complex nuclear processes, such as those in stellar nucleosynthesis or reactor fuel cycles, involve networks of reactions. Categorically:

Definition 6.2 (Reaction Network Category)**.** A nuclear reaction network is a free category $\mathcal{N}$ generated by:

- Objects: nuclides
- Morphisms: individual reactions

with composition representing sequential reactions.

Functors $\mathcal{N} \to$ **Rates** assign rate constants to reactions, enabling kinetic modeling.

6.3 Quantum Extensions

At the quantum level, nuclear states are vectors in Hilbert spaces, and processes are described by S-matrix elements. The categorical perspective suggests:

- **Nucl** embeds into **Hilb** (category of Hilbert spaces)
- Fission amplitudes become morphisms in **Hilb**
- The binding functor extends to a functor from **Hilb** to spectral data

6.4 Connection to Existing Categorical Physics

This work connects to the broader program of categorical and functorial physics:

- **Topological Quantum Field Theory (TQFT)**: TQFTs are functors from cobordism categories; our binding functor is analogous.
- **Categorical Quantum Mechanics**: The monoidal structure on **Nucl** parallels that in categorical quantum mechanics.
- **Coecke-Kissinger Framework**: Graphical calculi for monoidal categories could provide diagrammatic tools for nuclear physics.

6.5 Verification and Proof Assistance

The type-safe Haskell implementation of Section 5.4 demonstrates that conservation laws can be enforced at compile time. Extensions to dependent type systems (Agda, Idris, Lean) would enable:

- Machine-verified proofs of conservation
- Certified nuclear reaction databases
- Formally verified reactor simulations

7 Discussion

7.1 Novelty of This Work

The physics of nuclear fission has been understood for nearly a century. What is new here is the systematic categorical reformulation:

1. **Structural Perspective**: Viewing nuclear states as objects and processes as morphisms reveals compositional structure obscured in traditional presentations.
2. **Functorial Binding Energy**: Treating binding energy as a functor, not merely a function, captures how processes transform energy.
3. **Conservation as Structure**: Encoding conservation laws in the category structure makes illegal transitions unrepresentable.
4. **Coalgebraic Dynamics**: The coalgebraic formulation of chain reactions provides a principled framework for branching processes.
5. **Type-Safe Implementation**: The Haskell code demonstrates that categorical abstractions yield real computational benefits.

7.2 Limitations

This work has several limitations:

1. **SEMF Approximation**: The semi-empirical mass formula is an approximation; real binding energies require tabulated data or shell-model calculations.
2. **Simplified Chain Reactions**: Real reactor physics involves spatial neutron transport, energy spectra, and complex feedback mechanisms not captured here.
3. **Classical Treatment**: Quantum aspects of fission (tunneling, resonances, level densities) are not addressed categorically.

7.3 Future Work

Several directions merit further investigation:

1. **Quantum Categorical Fission**: Extending the framework to quantum-mechanical descriptions of fission.
2. **Higher Categories**: Using 2-categories or $(\infty, 1)$-categories to model higher-order nuclear processes.

3. **Machine-Verified Physics**: Implementing the framework in Agda or Lean for machine-verified nuclear physics.

4. **Applied Reactor Modeling**: Integrating the categorical framework with production reactor simulation codes.

8 Conclusion

We have presented a categorical reformulation of nuclear fission physics, treating nuclear configurations as objects in a symmetric monoidal category **Nucl**, fission processes as morphisms, binding energy as a functor $\mathcal{B} : \mathbf{Nucl} \to \mathbf{Energy}$, conservation laws as structural constraints, and chain-reaction dynamics as coalgebras.

The Haskell implementation demonstrates that these categorical abstractions are not merely theoretical niceties but translate into executable, type-safe code. Conservation laws become compile-time guarantees; functoriality ensures consistent energy accounting; coalgebraic structure enables principled modeling of branching dynamics.

While the underlying physics is well-established, this structural reformulation offers new perspectives and opens connections to the broader program of categorical physics. The framework provides a template for similar treatments of other nuclear processes (fusion, decay, nucleosynthesis) and other domains of physics amenable to categorical analysis.

The synthesis of category theory, nuclear physics, and typed functional programming exemplifies how modern mathematical and computational tools can illuminate even the most classical of physical theories.

Acknowledgments

The author thanks the developers of Haskell and the categorical physics community for creating the intellectual ecosystem in which this work was developed.

References

L. Meitner and O. R. Frisch. Disintegration of uranium by neutrons: A new type of nuclear reaction. *Nature*, 143:239–240, 1939.

O. Hahn and F. Strassmann. Über den Nachweis und das Verhalten der bei der Bestrahlung des Urans mittels Neutronen entstehenden Erdalkalimetalle. *Naturwissenschaften*, 27:11–15, 1939.

E. Fermi. The development of the first chain reacting pile. *Proceedings of the American Philosophical Society*, 90(1):20–24, 1946.

S. Mac Lane. *Categories for the Working Mathematician*. Springer-Verlag, 1971.

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

J. J. M. M. Rutten. Universal coalgebra: A theory of systems. *Theoretical Computer Science*, 249(1):3–80, 2000.

B. Coecke and A. Kissinger. *Picturing Quantum Processes: A First Course in Quantum Theory and Diagrammatic Reasoning*. Cambridge University Press, 2017.

J. C. Baez and M. Stay. Physics, topology, logic and computation: A Rosetta Stone. In B. Coecke, editor, *New Structures for Physics*, pages 95–172. Springer, 2011.

K. S. Krane. *Introductory Nuclear Physics*. John Wiley & Sons, 1988.

C. F. von Weizsäcker. Zur Theorie der Kernmassen. *Zeitschrift für Physik*, 96:431–458, 1935.

H. A. Bethe and R. F. Bacher. Nuclear physics A. Stationary states of nuclei. *Reviews of Modern Physics*, 8:82–229, 1936.

B. Jacobs. *Introduction to Coalgebra: Towards Mathematics of States and Observation*. Cambridge University Press, 2016.

A Complete Haskell Module

The complete Haskell implementation is available in the accompanying code repository at:

`https://github.com/MagnetonIO/functorial-fission`

B Table of Nuclide Data

Nuclide	A	Z	B (MeV)
^{1}n	1	0	0
^{1}H	1	1	0
^{4}He	4	2	28.30
^{56}Fe	56	26	492.26
^{92}Kr	92	36	783.18
^{141}Ba	141	56	1174.20
^{235}U	235	92	1783.87
^{236}U	236	92	1790.42
^{238}U	238	92	1801.69
^{239}Pu	239	94	1806.92

Table 2: Binding energies for selected nuclides (experimental values).

C Categorical Definitions Summary

For reference, we summarize the key categorical definitions:

Definition C.1 (Category)**.** A *category* $\mathbf{C}$ consists of:

- A collection $\mathrm{ob}(\mathbf{C})$ of objects
- For each pair of objects A, B, a set $\mathrm{Hom}(A, B)$ of morphisms
- Composition $\circ : \mathrm{Hom}(B, C) \times \mathrm{Hom}(A, B) \to \mathrm{Hom}(A, C)$
- Identity morphisms $\mathrm{id}_A \in \mathrm{Hom}(A, A)$

satisfying associativity and identity laws.

Definition C.2 (Functor)**.** A *functor* $F : \mathbf{C} \to \mathbf{D}$ consists of:

- A map $F : \mathrm{ob}(\mathbf{C}) \to \mathrm{ob}(\mathbf{D})$

- For each $A, B \in \mathbf{C}$, a map $F : \mathrm{Hom}_{\mathbf{C}}(A, B) \to \mathrm{Hom}_{\mathbf{D}}(F(A), F(B))$

preserving identity and composition.

Definition C.3 (Natural Transformation)**.** A *natural transformation* $\eta : F \Rightarrow G$ between functors $F, G : \mathbf{C} \to \mathbf{D}$ consists of morphisms $\eta_A : F(A) \to G(A)$ for each $A \in \mathbf{C}$, such that for all $f : A \to B$:

$$\eta_B \circ F(f) = G(f) \circ \eta_A \tag{41}$$

Definition C.4 (Coalgebra)**.** For an endofunctor $F : \mathbf{C} \to \mathbf{C}$, an *$F$-coalgebra* is a pair (A, α) where $A \in \mathbf{C}$ and $\alpha : A \to F(A)$.