Categorical Structure of Atomic Normalization for SLP-Definable Deterministic Transitions

1. Introduction

1.1 From Atomic Normalization to Structural Theory

Atomic representations of computational transitions provide a canonical structural level for the analysis of deterministic computations. In a series of preceding works we developed a structural framework for studying such representations in the context of deterministic transitions definable by straight-line programs (SLPs).

In the first paper of the series [1], we established the existence of **atomic normal forms** for SLP-definable deterministic transitions. The main result of that work shows that every deterministic transition representable by a straight-line program admits a normalization procedure producing a finite directed acyclic graph whose vertices correspond to primitive operations of a fixed signature. The resulting atomic representation is canonical up to interface-preserving isomorphism. This provides a uniform atomic level at which deterministic transitions can be studied independently of the syntactic structure of the original SLP.

In the second paper [2], we investigated the **algebraic structure** of the space of atomic forms. Treating atomic forms as directed acyclic graphs equipped with ordered input ports and ordered interfaces, we showed that the resulting space admits natural structural operations. In particular, atomic forms can be composed via interface gluing, yielding a strictly associative operation with identity. Furthermore, structural embeddings between atomic forms induce a partial order on the corresponding factor space modulo interface-preserving isomorphism. Together with several local structural reduction rules, these constructions show that the space of atomic forms possesses a rich internal algebraic organization.

The third paper of the series [3] introduced **compositional structural metrics** on atomic forms. These metrics quantify structural properties such as length, depth, and parallelism while remaining compatible with the algebraic operations defined on atomic forms. The resulting class of compositional structural metrics provides a quantitative perspective on the structural space introduced in [2], revealing trade-offs between different structural measures and establishing closure properties of the metric class.

Taken together, these works establish the existence, algebraic organization, and quantitative analysis of atomic representations of deterministic transitions. However, an additional structural aspect remains to be clarified. The constructions developed in the previous papers strongly suggest the presence of an underlying **categorical structure** governing atomic forms and their relation to SLP transitions. In particular, the normalization procedure introduced in [1] behaves in many respects like a canonical structural map from the space of SLP transitions to the space of atomic forms.

The purpose of the present paper is to make this observation precise.

We refer to the resulting framework as the **structural theory of atomic representations of deterministic transitions**.

This paper is the fourth part of a series devoted to the structural theory of atomic computational forms.

Novelty and Position of the Framework.

Although atomic forms are represented by directed acyclic graphs, the framework developed in this series differs from classical DAG, circuit, or dataflow representations in several essential aspects.

First, atomic forms introduce **explicit ordered vertex input ports** as structural objects of the graph.
Second, atomic forms incorporate **ordered input and output interfaces**, enabling canonical composition of computational structures.
Third, the **atomic normalization procedure** produces canonical atomic representations of SLP-definable transitions that are independent of the syntactic structure of the original program.
Finally, the framework supports **algebraic, metric, and categorical analysis directly at the atomic structural level**.
To the best of the author's knowledge, this combination of canonical normalization, port-structured computational graphs, compositional structural metrics, and categorical reflection has not previously been formulated in the literature.

Author Name Sergei Zubov
zubov369@gmail.com

1.2 Motivation for a Categorical Perspective

Several features of the framework developed in [1–3] naturally point toward a categorical interpretation.
First, both SLP transitions and atomic forms admit natural notions of **composition**. In the case of atomic forms, composition arises through interface gluing, as shown in [2]. For SLP transitions, composition corresponds to sequential execution of straight-line programs. These operations behave associatively and admit identity elements, suggesting an interpretation in terms of morphism composition.
Second, both levels admit natural notions of **structure-preserving mappings**. In the atomic setting these are given by structural embeddings preserving vertex labels, port orderings, and interfaces. Similar structural mappings exist between SLP representations. Such mappings behave functorially with respect to composition.
Third, the **normalization procedure** introduced in [1] systematically transforms SLP transitions into atomic forms. The compatibility of normalization with structural composition and embeddings, established in [2], suggests that normalization behaves as a structure-preserving transformation between the two structural levels.
These observations motivate the study of the categorical structure underlying atomic forms and SLP transitions. A categorical perspective provides a natural language for describing canonical constructions, structural invariants, and universal properties that are not easily expressed within purely algebraic frameworks.

1.3 Goals of the Present Work

The main objective of this paper is to identify and formalize the categorical structure underlying the theory of atomic forms developed in the previous works.
More specifically, we pursue the following goals.
First, we introduce a **category of atomic forms** whose objects are atomic forms over a fixed signature and whose morphisms are structural embeddings preserving vertex labels, port orderings, and interfaces. We show that these morphisms compose naturally and admit identities, yielding a well-defined category.

Second, we define a corresponding **category of SLP-definable deterministic transitions**. Objects of this category are SLP transitions with a fixed interface type, and morphisms are structure-preserving mappings between their graph representations.
Third, we show that the **atomic normalization procedure** introduced in [1] extends naturally to a functor between these two categories. This functor maps SLP transitions to their atomic representatives and preserves structural composition.
Fourth, we establish a **universal property** characterizing atomic normalization. Informally, atomic forms constitute the canonical structural level through which structural mappings from SLP transitions factor uniquely. In categorical terms, this property can be expressed by showing that the category of atomic forms forms a reflective subcategory of the category of SLP transitions.
This categorical characterization provides a conceptual explanation for the structural properties observed in [1–3]. In particular, it clarifies why atomic forms serve as a canonical structural representation for deterministic transitions.

1.4 Contribution of the Paper

The principal contributions of this paper can be summarized as follows.

1. We define the **category of atomic forms** associated with a fixed signature and interface type.
2. We introduce the **category of SLP-definable deterministic transitions** and describe its structural morphisms.
3. We show that the **normalization procedure** from SLP transitions to atomic forms defines a functor between these categories.
4. We establish a **universal characterization** of normalization, showing that the category of atomic forms is a reflective subcategory of the category of SLP transitions.
5. We derive several **structural consequences** of this categorical perspective, including the compatibility of normalization with structural embeddings and compositional metrics.

1.5 Organization of the Paper

The remainder of the paper is organized as follows.
Section 2 introduces the category of atomic forms and defines its morphisms and composition.
Section 3 develops the category of SLP-definable deterministic transitions.
Section 4 shows that atomic normalization extends to a functor between these categories.
Section 5 establishes the universal property of normalization and proves the reflection theorem.
Section 6 discusses structural consequences of the categorical framework and its relation to the algebraic and metric structures developed in [2] and [3].

2. Category of Atomic Forms

In this section we introduce the categorical structure underlying atomic forms. The goal is to formalize the structural mappings between atomic forms and to show that they naturally form a category.
Throughout this section we fix a finite signature Σ and a fixed interface arity nnn. All atomic forms considered below therefore have interface type

$$X^n \to X^n.$$

The notion of atomic form used here follows the definition introduced in [2]: an atomic form is a finite directed acyclic graph whose vertices are labeled by operations of the signature, whose argument positions are represented by ordered input ports, and whose external boundary is given by ordered input and output interface ports.
We now introduce morphisms between such structures.

2.1 Objects

The objects of the category are atomic forms over the fixed signature Σ with interface type

$$X^n \to X^n.$$

Recall that an atomic form

$$A = (V(A), Ports_A, E(A), \ell_A, I(A), O(A))$$

consists of

- a finite set $V(A)$ of vertices,
- vertex input ports $Ports_A(v) = \{(v,1), \dots, (v,k)\}$ determined by the arity of the operation labeling v,
- a set of directed edges
$$E(A) \subseteq (V(A) \cup I(A)) \times (Ports_A \cup O(A)),$$
- a labeling function $\ell_A: \mathrm{V(A)} \to \Sigma$,
- ordered input interface ports
$$I(A) = ((in,1), \dots, (in,n)),$$
- ordered output interface ports
$$O(A) = ((out,1), \dots, (out,n)).$$

The induced directed graph on vertices is required to be acyclic.
Objects are considered modulo interface-preserving isomorphism as defined in [2].

2.2 Structural Morphisms

We now define the morphisms between atomic forms.

Definition 2.1 (Structural Morphism)

Let A and B be atomic forms with the same interface type.
A **structural morphism**

$$\phi: A \to B$$

is given by an injective mapping

$$\phi: V(A) \to V(B)$$

satisfying the following conditions.

Label preservation

For every vertex $v \in V(A)$,

$$\ell_A: (v) = \ell_B(\phi(v))$$

Thus operations labeling vertices are preserved under the mapping.

Port correspondence

For every vertex $v \in V(A)$ with arity k,

$$Ports_A(v) = \{(v,1), \dots, (v,k)\}$$

corresponds to

$$Ports_B(v) = \{(\phi(v),1), \dots, (\phi(v), k)\}.$$

In particular, the index i of each input port is preserved.

Edge preservation

Edges must be preserved with respect to ports.
For every edge

$$(s, t) \in E(A)$$

we require

$$(\hat{\varphi}(s), \hat{\varphi}(t)) \in E(B),$$

where $\hat{\varphi}$ is the canonical extension of φ defined by

- $\hat{\varphi}(v) = \varphi(v)$ for $v \in V(A)$;
- $\hat{\varphi}((v, i)) = (\varphi(v), i)$ for vertex input ports;
- $\hat{\varphi}(x) = x$ for interface ports $x \in I(A) \cup O(A)$.

Thus structural morphisms preserve the exact attachment of edges to ports.

Interface preservation

The interface of the atomic form is fixed under morphisms:

$$I(A) = I(B),\ O(A) = O(B).$$

In particular, interface ports cannot be permuted or renamed.
Structural morphisms therefore represent **embeddings of one atomic form into another** that preserve labels, port ordering, edges, and the external interface.

2.3 Composition of Morphisms

Structural morphisms compose naturally via composition of their underlying vertex mappings.
Let

$$\phi: A \to B,\ \psi: B \to C$$

be structural morphisms.
We define their composition

$$\psi \circ \phi: A \to C$$

by

$$(\psi \circ \phi)(v) = \psi(\phi(v)).$$

Since both mappings are injective and preserve labels, ports, edges, and interfaces, the composed mapping satisfies the same structural conditions.

Proposition 2.1

The composition of two structural morphisms is again a structural morphism.

Proof

Label preservation follows immediately from the preservation properties of ϕ and ψ.

For each vertex v, the port ordering is preserved because both mappings preserve the port indices.
Edge preservation holds because if

$$(s,t) \in E(A),$$

then

$$(\hat{\varphi}(s), \hat{\varphi}(t)) \in E(B),$$

and applying the extension of ψ yields

$$(\hat{\psi}(\hat{\varphi}(s)), \hat{\psi}(\hat{\varphi}(t))) \in E(C).$$

But this is exactly

$$(\widehat{\psi \circ \phi}(s), \widehat{\psi \circ \phi}(t)).$$

Interface ports remain fixed under both mappings, so they remain fixed under composition.
Thus the composition is again a structural morphism. ■

2.4 Identity Morphisms
For every atomic form A we define the identity morphism

$$id_A : A \to A$$

by

$$id_A(v) = v$$

for all $v \in V(A)$.
The extension of id_A to ports and interface elements is also the identity.

Proposition 2.2
The mapping id_A is a structural morphism.
Proof
The identity mapping preserves labels, ports, edges, and interface ports trivially. ■

2.5 Category of Atomic Forms
We now state the main result of this section.
Theorem 2.1
Atomic forms over a fixed signature Σ and interface type $X^n \to X^n$, together with structural morphisms defined above, form a category.
We denote this category by

$$\boldsymbol{Atom}_{\Sigma}.$$

Proof
The objects of the category are atomic forms.
Morphisms are structural morphisms between atomic forms.
Proposition 2.1 shows that morphisms are closed under composition.
Proposition 2.2 shows that identity morphisms exist.
Associativity of morphism composition follows directly from associativity of function composition on vertex mappings.
Therefore the axioms of a category are satisfied. ■

2.6 Interpretation

The category $\boldsymbol{Atom}_{\Sigma}$ captures the structural relationships between atomic forms.
Morphisms represent embeddings of atomic computation structures preserving all structural information:

- operation labels,
- argument ordering,
- edge attachment to ports,
- external interfaces.

This categorical perspective provides a natural structural setting for studying atomic representations of deterministic transitions. In subsequent sections we relate this category to the category of SLP-definable transitions and show that the normalization procedure introduced in [1] defines a canonical functor between them.

3. Category of SLP-Definable Transitions

In this section we introduce the categorical structure associated with deterministic transitions definable by straight-line programs (SLPs). This structure provides the source category for the normalization functor introduced later.
The constructions developed here parallel those of the category of atomic forms introduced in Section 2, but operate at the level of SLP representations.
Throughout the section we fix a finite signature Σ and a fixed interface arity nnn. All transitions therefore have interface type

$$X^n \to X^n.$$

3.1 SLP-Definable Deterministic Transitions

We briefly recall the notion of an SLP-definable transition introduced in [1].
A **straight-line program (SLP)** over a signature Σ is a finite sequence of assignments computing values using the operations of Σ. Each assignment produces a new value by applying an operation of the signature to previously computed values.
Such programs naturally induce deterministic transitions between tuples of values.
Formally, an SLP-definable transition can be represented by a finite directed graph

$$H = (V(H), E(H), \ell_H, I(H), O(H))$$

where

- $V(H)$ is a finite set of computational vertices,
- $E(H)$ is a set of directed edges representing data dependencies,
- $\ell_H: V(H) \to \Sigma$ labels vertices with operations,
- $I(H)$ is the ordered list of input interface ports,
- $O(H)$ is the ordered list of output interface ports.

The graph encodes the data dependencies of the straight-line program: edges connect outputs of previously computed vertices (or inputs) to argument positions of subsequent operations.
Because SLPs contain no loops or recursion, the induced graph is acyclic.
The semantics of such a structure is the deterministic function

$$[\![H]\!]: X^n \to X^n$$

computed by evaluating the graph in a topological order.

3.2 Objects

The objects of the category introduced in this section are SLP-definable deterministic transitions with fixed interface type $X^n \to X^n$.
Each object is represented by a dependency graph

$$H = (V(H), E(H), \ell_H, I(H), O(H))$$

as described above.
Objects are considered modulo interface-preserving isomorphism of their underlying graphs. Such isomorphisms preserve

- vertex labels,
- edge structure,
- ordering of arguments,
- ordering of interface ports.

This equivalence reflects the fact that SLPs differing only by renaming of intermediate variables represent the same structural transition.

3.3 Structural Morphisms

We now introduce morphisms between SLP transitions.

Definition 3.1 (Structural Morphism)

Let H_1 and H_2 be SLP-definable transitions with the same interface type.
A **structural morphism**

$$\phi: H_1 \to H_2$$

is an injective mapping

$$\phi: V(H_1) \to V(H_2)$$

satisfying the following conditions.

Label preservation

For every vertex $v \in V(H_1)$,

$$\ell_{H1}(v) = \ell_{H2}(\phi(v)).$$

Thus operations labeling vertices are preserved.

Dependency preservation

If

$$(u, v) \in E(H_1)$$

then

$$(\phi(u), \phi(v)) \in E(H_2).$$

This condition ensures that the data dependency structure of the transition is preserved.

Argument position preservation
The ordering of arguments of each operation must be preserved. If an edge of H_1 corresponds to the i-th argument of an operation, the corresponding edge in H_2 must correspond to the same argument position.

Interface preservation
The interface of the transition is fixed:

$$I(H_1) = I(H_2), O(H_1) = O(H_2)$$

Thus morphisms do not permute or rename interface ports.

Structural morphisms therefore represent embeddings of one transition structure into another while preserving the operational semantics of the computation.

3.4 Composition of Morphisms

Let

$$\phi: H_1 \to H_2, \phi: H_2 \to VH_3$$

be structural morphisms.
Their composition is defined as

$$(\psi \circ \phi)(v) = \psi(\phi(v)).$$

Proposition 3.1
The composition of structural morphisms is again a structural morphism.
Proof
Label preservation follows directly from the preservation properties of ϕ and ψ.
If

$$(u, v) \in E(H_1)$$

Then

$$(\phi(u), \phi(v)) \in E(H_2)$$

and

$$(\psi(\phi(u)), \psi(\phi(v))) \in E(H_3)$$

Thus edge preservation holds for the composition.
Preservation of argument ordering follows from the same property for ϕ and ψ.
Interface ports remain fixed under both mappings and therefore remain fixed under the composition.
Hence the composition is again a structural morphism. ■

3.5 Identity Morphisms

For every transition H we define the identity morphism

$$id_H : H \to H$$

by

$$id_H(v) = v$$

for all vertices $v \in V(H)$.

Proposition 3.2

The mapping id_H is a structural morphism.

Proof

The identity mapping preserves labels, edges, argument positions, and interface ports trivially. ■

3.6 Category of SLP Transitions

We now obtain the main result of this section.

Theorem 3.1

SLP-definable deterministic transitions over a fixed signature Σ and interface type $X^n \to X^n$, together with structural morphisms defined above, form a category.

We denote this category by

$$\boldsymbol{SLP}_{\Sigma}.$$

Proof

Objects are SLP-definable transitions.

Morphisms are structural morphisms between them.

Proposition 3.1 shows that morphisms are closed under composition.

Proposition 3.2 shows that identity morphisms exist.

Associativity of morphism composition follows from associativity of function composition.

Therefore the axioms of a category are satisfied. ■

3.7 Relation to Atomic Forms

The category $\boldsymbol{SLP}_{\Sigma}$ describes deterministic transitions at the level of their straight-line program representations.

In contrast, the category $\boldsymbol{Atom}_{\Sigma}$ introduced in Section 2 describes transitions at the level of atomic computational structures.

The normalization procedure introduced in [1] systematically transforms SLP transitions into atomic forms. In the next section we show that this transformation extends naturally to a functor between the categories

$$Norm : \boldsymbol{SLP}_{\Sigma} \to \boldsymbol{Atom}_{\Sigma}$$

This functor provides the categorical bridge between SLP transitions and their canonical atomic representations.

4. Normalization as a Functor

In this section we show that the atomic normalization procedure introduced in [1] naturally extends to a functor between the category of SLP-definable transitions and the category of atomic forms constructed in the previous sections.

Intuitively, normalization replaces the syntactic structure of a straight-line program with its canonical atomic representation while preserving the structural relationships between transitions. The compatibility properties established in [2] allow this transformation to be interpreted categorically.

4.1 Atomic Normalization

We briefly recall the normalization procedure introduced in [1].
For every SLP-definable deterministic transition

$$H$$

there exists an atomic form A such that over the signature Σ, the normalization procedure constructs an atomic form

$$Norm(H)$$

whose vertices correspond to primitive operations and whose structure represents the same deterministic computation.
The construction proceeds by expanding each operation in the SLP graph into its atomic computational structure while preserving data dependencies and interface ports.
The result is a directed acyclic graph

$$Norm(H) = (V_N, Ports_N, E_N, \ell_N, I_N, O_N)$$

satisfying the definition of an atomic form introduced in [2].

The central structural result underlying the entire framework is the following normalization theorem established in the first paper of the series [1].

Theorem 4.1 (Atomic Normalization Theorem, Zubov 2026)
For every SLP-definable deterministic transition H there exists an atomic form $Norm(H)$ such that

$$[\![Norm(H)]\!] = [\![H]\!].$$

Moreover, the resulting atomic form is unique up to interface-preserving isomorphism.
This theorem was established in [1].

4.2 Action on Objects

We now define the action of normalization on objects of the category $\boldsymbol{SLP}_{\Sigma}$.
Definition 4.1
Let H be an object of $\boldsymbol{SLP}_{\Sigma}$.
We define

$$Norm(H)$$

to be the atomic form obtained by applying the normalization procedure of [1] to the SLP transition H.
Since atomic forms are considered modulo interface-preserving isomorphism, the normalization is well defined as an object of the category

$$\boldsymbol{Atom}_{\Sigma}.$$

Thus normalization defines a mapping

$$Norm: Obj(\boldsymbol{SLP}_{\Sigma}) \to Obj(\boldsymbol{Atom}_{\Sigma}).$$

4.3 Action on Morphisms
We now extend normalization to morphisms.
Let

$$\phi: H_1 \to H_2$$

be a structural morphism between SLP transitions.
Such a morphism is an injective mapping of vertices preserving labels, dependencies, argument ordering, and interface ports.
The normalization procedure expands each vertex of an SLP transition into its corresponding atomic structure. Because the structural morphism ϕ\phiϕ preserves the dependency structure, the expansion of vertices in H_1 is mapped naturally into the expansion of vertices in H_2.
This induces a structural morphism between the corresponding atomic forms.

Definition 4.2
Let

$$\phi: H_1 \to H_2$$

be a structural morphism in $\boldsymbol{SLP}_{\Sigma}$.
We define

$$Norm(\phi): Norm(H_1) \to Norm(H_2)$$

to be the structural morphism obtained by mapping the atomic expansion of each vertex v in H_1 to the atomic expansion of the vertex $\phi(v)$ in H_2.
This mapping preserves

- operation labels,
- port indices,
- edge attachments,
- interface ports.

Therefore it is a structural morphism in the category $\boldsymbol{Atom}_{\Sigma}$.

4.4 Preservation of Identity
We now verify that normalization preserves identity morphisms.
Proposition 4.1
For every SLP transition H,

$$Norm(id_H) = id_{Norm\,(H)}.$$

Proof
The identity morphism

$$id_H : H \to H$$

maps each vertex to itself.
Under normalization, the atomic expansion of each vertex is mapped to itself as well. Therefore the induced mapping on the normalized structure is the identity mapping.
Thus

$$Norm(id_H) = id_{Norm\,(H)}.$$

■

4.5 Preservation of Composition
We now show that normalization preserves morphism composition.
Proposition 4.2
Let

$$\phi : H_1 \to H_2, \psi : H_2 \to H_3,$$

be structural morphisms in $\boldsymbol{SLP}_{\Sigma}$.
Then

$$Norm(\psi \circ \phi) = Norm(\psi) \circ Norm(\phi).$$

Proof
The morphism

$$\psi \circ \phi$$

maps vertices of H_1 to vertices of H_3.
Under normalization, the expansion of each vertex of H_1 is mapped to the expansion of its image in H_3.
This is equivalent to first mapping expansions from H_1 to H_2 via $Norm(\phi)$, and then mapping them from H_2 to H_3 via $Norm(\psi)$.
Therefore the induced mapping on normalized forms satisfies

$$Norm(\psi \circ \phi) = Norm(\psi) \circ Norm(\phi).$$

■

4.6 The Normalization Functor
We now obtain the main result of this section.
Theorem 4.2
The normalization procedure defines a functor

$$Norm : \boldsymbol{SLP}_{\Sigma} \to \boldsymbol{Atom}_{\Sigma}.$$

Proof
Normalization maps objects of $\boldsymbol{SLP}_{\Sigma}$ to objects of $\boldsymbol{Atom}_{\Sigma}$.

By Definition 4.2 it maps morphisms of $\boldsymbol{SLP}_{\Sigma}$ to morphisms of $\boldsymbol{Atom}_{\Sigma}$.
Proposition 4.1 shows that identity morphisms are preserved.
Proposition 4.2 shows that morphism composition is preserved.
Therefore normalization satisfies the axioms of a functor.
■

4.7 Interpretation
The functor

$$Norm: \boldsymbol{SLP}_{\Sigma} \to \boldsymbol{Atom}_{\Sigma}$$

maps syntactic representations of deterministic computations to their canonical atomic forms.
This functor preserves structural embeddings and sequential composition of transitions.
Consequently, the category of atomic forms provides a canonical structural image of the category of SLP-definable transitions.
In the next section we show that this relationship satisfies a universal property characterizing atomic normalization as a categorical reflection.

5. Universal Property of Atomic Normalization
In the previous section we established that the normalization procedure defines a functor

$$Norm: \boldsymbol{SLP}_{\Sigma} \to \boldsymbol{Atom}_{\Sigma}.$$

This functor maps SLP-definable transitions to their atomic representations while preserving structural morphisms.
In this section we show that normalization satisfies a **universal property**. Informally, atomic forms constitute a canonical structural level through which structural morphisms from SLP transitions factor uniquely.
This property provides a categorical explanation for the canonical role played by atomic forms in the structural theory developed in [1–3].

5.1 Atomic Forms as SLP Transitions
Every atomic form can naturally be interpreted as an SLP-definable transition.
Indeed, an atomic form already represents a finite acyclic computation graph whose vertices correspond to primitive operations of the signature. Such a structure can be interpreted directly as a straight-line computation.
Thus there exists a natural embedding

$$J: \boldsymbol{Atom}_{\Sigma} \to \boldsymbol{SLP}_{\Sigma}$$

which maps each atomic form to the corresponding SLP transition represented by the same computational graph.

Definition 5.1 (Embedding Functor)
For an atomic form

$$A = (V(A), Ports_A, E(A), \ell_A, I(A), O(A)),$$

we define

$$J(A)$$

to be the SLP transition represented by the same computational graph.
For a structural morphism

$$\psi: A_1 \to A_2,$$

we define

$$J(\psi)$$

to be the same vertex mapping interpreted as a structural morphism of SLP transitions.

Proposition 5.1
The mapping J defines a functor

$$J: \boldsymbol{Atom}_{\Sigma} \to \boldsymbol{SLP}_{\Sigma}.$$

Proof
Objects are mapped to SLP transitions represented by the same graph.
Morphisms are preserved since structural embeddings between atomic forms preserve labels, dependencies, port orderings, and interface ports, which are exactly the structural properties required for morphisms between SLP transitions.
Identity morphisms and composition are preserved trivially.
Therefore J is a functor. ■

5.2 Essential Image of Normalization
The normalization procedure introduced in the first paper of this series associates to every SLP-definable transition a corresponding atomic form.
However, not every atomic form necessarily arises as the normalization of an SLP transition. In order to obtain a precise categorical relationship between the two structures, we therefore restrict attention to the class of atomic forms that occur as normalization results.
Definition 5.2 (Normal Atomic Form)
An atomic form A A is called **normal** if there exists an SLP-definable transition H such that

$$A \cong Norm(H).$$

We denote by

$$\boldsymbol{Atom}_{\Sigma}$$

the full subcategory of atomic forms consisting of normal atomic forms.
Thus every object of $\boldsymbol{Atom}_{\Sigma}$ is (up to interface-preserving isomorphism) the atomic normalization of some SLP transition.
This restriction allows the normalization functor to exhibit a universal property.

5.3 Properties of the Embedding Functor
Every atomic form can be interpreted as an SLP transition by viewing its directed acyclic graph as a straight-line evaluation procedure.
This interpretation defines a functor

$$I: \boldsymbol{Atom}_{\Sigma} \to \boldsymbol{SLP}_{\Sigma}$$

called the **embedding functor**.
On objects, $I(A)$ is the SLP transition obtained by evaluating the DAG underlying A.
On morphisms, I acts as the identity map on vertex embeddings.
Proposition 5.1
The mapping I defines a functor.
Proof.
Identity morphisms are preserved because the identity embedding of an atomic form corresponds to the identity embedding of the corresponding SLP transition.
Compositions are preserved because compositions of embeddings remain embeddings under the interpretation of atomic forms as SLP transitions.
Thus I is a well-defined functor.

5.4 Universal Factorization
The normalization procedure defines a functor

$$Norm: \boldsymbol{SLP}_{\Sigma} \to \boldsymbol{Atom}_{\Sigma}$$

Intuitively, normalization replaces each SLP operation by its canonical atomic expansion while preserving the dependency structure of the program.
Proposition 5.2 (Universal Factorization Property)
Let

$$H \in \boldsymbol{SLP}_{\Sigma}$$

and

$$A \in \boldsymbol{Atom}_{\Sigma}.$$

For every structural embedding

$$f: H \to I(A)$$

there exists a unique structural embedding

$$g: Norm(H) \to A$$

such that

$$f = I(g) \circ \eta H$$

where

$$\eta_H: H \to I(Norm(H))$$

is the canonical embedding induced by normalization.
Proof.
Since A is normal, there exists an SLP transition K such that

$$A \cong Norm(K).$$

The embedding $f: H \to I(A)$ therefore corresponds to an embedding of SLP transitions

$$H \to I(Norm(K)).$$

Because normalization expands each SLP operation into its canonical atomic structure without altering the dependency ordering, the embedding f uniquely extends to an embedding

$$Norm(H) \to Norm(K).$$

Transporting this embedding along the isomorphism $Norm(K) \cong A$ yields the desired morphism

$$g: Norm(H) \to A.$$

Uniqueness follows from the injectivity and structure-preserving properties of embeddings.

5.5 Reflection Theorem

We now obtain the main categorical result.

Theorem 5.1 (Reflection Theorem)

The normalization functor

$$Norm: \boldsymbol{SLP}_{\Sigma} \to \boldsymbol{Atom}_{\Sigma}$$

is left adjoint to the embedding functor

$$I: \boldsymbol{Atom}_{\Sigma} \hookrightarrow \boldsymbol{SLP}_{\Sigma}.$$

Equivalently, for every SLP transition H and every atomic form A there exists a natural bijection

$$Hom_{\boldsymbol{Atom}_{\Sigma}}(Norm(H), A) \;\cong\; Hom_{\boldsymbol{SLP}_{\Sigma}}(H, I(A)).$$

Proof.

The universal factorization property established in Proposition 5.2 provides a bijective correspondence between morphisms

$$H \to I(A)$$

and morphisms

$$Norm(H) \to A.$$

Naturality of this correspondence with respect to both arguments follows from the functoriality of normalization and the embedding functor.

Thus the pair $(Norm, I)$ forms an adjunction.

Since I is a full inclusion of the subcategory of normal atomic forms, the category $\boldsymbol{Atom}_{\Sigma}$ is a reflective subcategory of $\boldsymbol{SLP}_{\Sigma}$.

5.6 Interpretation

The reflection theorem provides a conceptual explanation for the structural role of atomic forms. Atomic forms constitute the **canonical structural level** for deterministic transitions definable by straight-line programs. Every such transition admits a canonical atomic representative, and structural mappings into atomic forms factor uniquely through this representation.

From the categorical perspective, normalization acts as a reflection that projects the category of SLP transitions onto its atomic subcategory.

This result explains why the algebraic and metric structures studied in [2] and [3] naturally arise at the atomic level.

6. Structural Consequences of the Categorical Framework

The categorical description developed in the previous sections clarifies the structural role of atomic forms within the theory of deterministic transitions. In particular, the reflection theorem established in Section 5 explains why the algebraic and metric structures introduced in the earlier papers naturally arise at the atomic level.
In this section we summarize several structural consequences of the categorical framework and relate them to the results obtained in [2] and [3].

6.1 Canonical Structural Representation

One of the main consequences of the reflection theorem is the canonical nature of atomic representations.
Recall that the normalization functor

$$Norm: \boldsymbol{SLP}_{\Sigma} \to \boldsymbol{Atom}_{\Sigma}$$

assigns to every SLP transition its atomic representative.

The adjunction

$$Norm \dashv J$$

established in Theorem 5.1 implies that atomic forms provide a universal representation of deterministic transitions within the category of SLP transitions.
More precisely, every structural morphism from an SLP transition into an atomic form factors uniquely through the normalized form. This means that the atomic representation captures exactly the structural information of the computation that is relevant for embeddings into atomic structures.
Consequently, atomic forms can be interpreted as **canonical structural representatives** of SLP-definable deterministic transitions.

6.2 Compatibility with Structural Composition

The algebraic structure of atomic forms introduced in [2] includes a composition operation defined via interface gluing.
Recall that for atomic forms A_1 and A_2 with compatible interfaces, their composition

$$A_2 \circ A_1$$

is obtained by identifying the output interface of A_1 with the input interface of A_2 and preserving the internal computational structure.
This operation was shown in [2] to be strictly associative and to admit an identity atomic form.
The categorical framework developed here provides a conceptual explanation for this structure.
Since SLP transitions compose via sequential execution, and normalization preserves composition (Section 4), the following compatibility holds:

$$Norm(H_2 \circ H_1) \cong Norm(H_2) \circ Norm(H_1).$$

Thus the algebraic composition of atomic forms arises naturally from the categorical structure induced by normalization.
In other words, the monoidal structure of atomic forms is inherited from the sequential composition of SLP transitions.

6.3 Compatibility with Structural Embeddings

In [2] we introduced structural embeddings between atomic forms and showed that they induce a partial order on the space of atomic forms modulo isomorphism.
From the categorical perspective developed here, these embeddings correspond precisely to the morphisms of the category

$$\boldsymbol{Atom}_{\Sigma}.$$

The reflection theorem implies that structural embeddings of SLP transitions are preserved by normalization. In particular, if

$$H_1 \preceq H_2$$

denotes a structural embedding between SLP transitions, then normalization yields a corresponding embedding between their atomic forms:

$$Norm(H_1) \preceq Norm(H_2).$$

Thus normalization is **monotone with respect to structural embeddings**.
This property explains why the partial order structure studied in [2] naturally appears at the atomic level.

6.4 Compatibility with Structural Metrics

The third paper of this series [3] introduced compositional structural metrics on atomic forms. These metrics assign numerical values to atomic structures while satisfying invariance, monotonicity, and subadditivity properties.
Examples include

- length metrics,
- depth metrics,
- measures of structural parallelism.

Because normalization preserves structural composition and embeddings, these metrics behave consistently with respect to the categorical framework developed here.
In particular, if MMM is a compositional structural metric defined on atomic forms, then for SLP transitions H_1 and H_2 we obtain

$$M(Norm(H_2 \circ H_1)) = M(Norm(H_2) \circ Norm(H_1)).$$

Similarly, structural embeddings preserved by normalization ensure that

$$H_1 \preceq H_2 \Rightarrow M(Norm(H_1)) \leq M(Norm(H_2)).$$

Thus compositional structural metrics are compatible with the categorical structure induced by normalization.

6.5 Structural Interpretation of the Theory

Combining the results of this paper with those of [1–3], we obtain a unified structural picture of deterministic transitions.

The theory developed across the four papers can be summarized as follows:

1. Every SLP-definable deterministic transition admits a canonical atomic representation ([1]).
2. The space of atomic forms carries a natural algebraic structure based on composition and structural embeddings ([2]).
3. Quantitative structural properties of atomic forms can be analyzed using compositional structural metrics ([3]).
4. The normalization procedure defines a reflection from the category of SLP transitions into the category of atomic forms (this paper).

Within this framework, atomic forms play the role of a canonical structural layer that captures the essential computational structure of deterministic transitions.

6.6 Scope of the Categorical Framework

The categorical framework developed in this paper focuses exclusively on sequential composition and structural embeddings.

Several extensions of this framework are possible but lie outside the scope of the present work. In particular, we do not consider

- parallel tensor structures on atomic forms,
- enriched categorical structures,
- higher categorical generalizations,
- algorithmic aspects of normalization.

The purpose of the present paper is to establish the fundamental categorical structure underlying atomic forms rather than to explore these extensions.

6.7 Summary

The categorical interpretation of atomic normalization provides a conceptual explanation for the structural properties observed in the previous papers of this series.

Atomic forms emerge as the canonical objects of a reflective subcategory within the category of SLP-definable deterministic transitions. This perspective unifies the algebraic and metric structures previously introduced and clarifies the role of normalization as a universal structural construction.

Together with the results of [1–3], the categorical framework presented here completes the structural theory of atomic representations of deterministic transitions.

7. Scope and Delimitations

The categorical framework developed in this paper provides a structural characterization of atomic representations of deterministic transitions. While the results obtained here clarify the universal role of atomic forms within the theory of SLP-definable transitions, it is important to explicitly state the scope of the present work and the limitations of the framework.

The goal of this section is therefore to delineate the precise boundaries of the theory established across the series of papers [1–4].

7.1 Basis Relativity

All constructions in the present theory are defined relative to a fixed finite signature Σ.
The signature specifies the primitive operations available for constructing both SLP transitions and atomic forms. Consequently, the normalization procedure, the algebraic structure of atomic forms, and the categorical framework introduced in this paper all depend on the choice of Σ.
In particular, different signatures may induce different atomic representations of the same computational function. The theory therefore studies structural properties of computations relative to a fixed computational basis rather than providing a basis-independent notion of atomicity.
This basis-relativity is inherent to the construction of atomic forms and does not affect the internal consistency of the theory.

7.2 Representation Scope

The results established in this series apply specifically to **deterministic transitions definable by straight-line programs**.
Straight-line programs describe computations consisting of a finite sequence of assignments without branching, loops, or recursion. The corresponding dependency graphs are therefore finite directed acyclic graphs.
The normalization procedure and the categorical constructions developed in this paper rely on this acyclic structure.
Computational models involving control flow, dynamic branching, or recursive evaluation fall outside the scope of the present framework.

7.3 Atomic Forms and Normalization

Although every SLP-definable deterministic transition admits an atomic representation, the converse statement is not claimed in this work.
In particular, not every atomic form necessarily arises as the normalization of some SLP transition. The category of atomic forms therefore contains objects that do not correspond directly to normalized SLP transitions.
The normalization functor

$$Norm: \boldsymbol{SLP}_{\Sigma} \to \boldsymbol{Atom}_{\Sigma}$$

should therefore be interpreted as embedding the structural information of SLP transitions into a larger structural space of atomic forms.
The reflection theorem established in Section 5 characterizes this relationship precisely.

7.4 Reduction Versus Normalization

The structural reduction rules introduced in [2] provide local simplifications of atomic forms. These rules include operations such as dead-node elimination, contraction of linear chains, and folding of common substructures.
These reductions preserve the semantics of the represented computation and decrease certain structural measures of the atomic form.

However, these reduction rules do **not** define a normalization procedure in the categorical sense.
In particular, the present work does not establish

- termination of reduction sequences,
- confluence of reductions,
- existence of canonical reduced representatives.

The atomic normalization procedure introduced in [1] therefore remains the primary canonical construction in the theory.

7.5 Quantitative Analysis

The compositional structural metrics introduced in [3] provide quantitative measures of atomic structures.
These metrics capture structural properties such as

- computational length,
- evaluation depth,
- degrees of parallelism.

While the categorical framework developed in the present paper explains why such metrics naturally live on atomic forms, the detailed study of quantitative invariants is carried out separately in [3].
The present paper therefore focuses exclusively on the structural and categorical aspects of the theory.

7.6 Possible Extensions

Several natural extensions of the present framework may be considered in future work.
These include, for example:

- the introduction of tensor structures representing parallel composition of atomic forms,
- enriched categorical formulations incorporating quantitative metrics,
- algorithmic analysis of normalization procedures,
- extensions of the framework to models with control flow.

Exploring these directions would require additional constructions beyond the structural framework established in this series.

7.7 Completion of the Structural Program

Taken together, the four papers of this series establish a coherent structural theory of atomic representations of deterministic transitions.
The results can be summarized as follows.
First, the existence of atomic representations for SLP-definable transitions was established in [1].
Second, the algebraic structure of the space of atomic forms was developed in [2].
Third, compositional structural metrics on atomic forms were introduced and studied in [3].
Finally, the present paper shows that atomic normalization admits a categorical characterization as a reflection between the category of SLP transitions and the category of atomic forms.
Within this framework, atomic forms emerge as the canonical structural level for representing deterministic computations.
The categorical perspective developed here therefore completes the structural theory of atomic representations initiated in the preceding works.

8. Conclusion

The purpose of this paper was to provide a categorical interpretation of atomic normalization for deterministic transitions defined by straight-line programs.

In the first paper of the series it was shown that every SLP-definable deterministic transition admits an atomic representation in the form of a directed acyclic computational structure. The second paper investigated the internal algebraic structure of the space of atomic forms, introducing composition, structural embeddings, and reduction rules. The third paper introduced compositional structural metrics that allow quantitative analysis of atomic structures.

The present work complements these results by providing a categorical framework that explains the structural role of atomic forms within the theory.

8.1 Categorical Characterization of Atomic Normalization

The central result of the paper is the categorical characterization of atomic normalization as a reflection.

More precisely, we considered two categories:

- the category $\boldsymbol{SLP}_{\Sigma}$ of SLP-definable deterministic transitions and structural embeddings between them, and
- the category $\boldsymbol{Atom}_{\Sigma}$ of atomic forms modulo interface-preserving isomorphism.

Within this setting we defined a normalization functor

$$Norm: \boldsymbol{SLP}_{\Sigma} \to \boldsymbol{Atom}_{\Sigma}$$

which assigns to every SLP transition its atomic representative.

We proved that this functor forms a reflection with respect to the inclusion functor

$$J: \boldsymbol{Atom}_{\Sigma} \to \boldsymbol{SLP}_{\Sigma}$$

This reflection theorem provides a conceptual explanation for the canonical role of atomic forms within the theory of deterministic transitions.

8.2 Structural Role of Atomic Forms

From the categorical point of view developed in this paper, atomic forms represent the canonical structural layer of deterministic computations.

The reflection property implies that every SLP transition admits a universal structural factorization through its atomic form. In other words, atomic forms capture precisely the structural information that is relevant for embeddings into atomic structures.

This perspective clarifies the results obtained in the previous papers of the series.

In particular:

- the monoidal composition of atomic forms studied in [2] arises naturally from the composition of SLP transitions;
- the partial order defined by structural embeddings corresponds to morphisms in the category of atomic forms;
- the compositional metrics introduced in [3] are naturally defined on the atomic level because normalization preserves structural composition and embeddings.

Thus the categorical framework developed here provides a unifying interpretation of the algebraic and metric structures introduced earlier.

8.3 Summary of the Structural Theory

Taken together, the four papers establish a coherent structural theory of atomic representations of deterministic transitions.

The main contributions can be summarized as follows.

First, the existence of atomic normal forms for SLP-definable deterministic transitions was established in [1].

Second, the internal algebraic structure of the space of atomic forms was investigated in [2], where composition, structural embeddings, and reduction rules were introduced.

Third, compositional structural metrics were defined and analyzed in [3], providing tools for quantitative analysis of atomic structures.

Finally, the present paper provides a categorical interpretation of atomic normalization and shows that atomic forms arise as objects of a reflective subcategory within the category of SLP transitions.

Together these results describe the structural landscape of atomic representations for deterministic computations.

8.4 Perspective

The categorical viewpoint developed here suggests several directions for future investigation.

One possible direction is the study of parallel or tensor structures on atomic forms. Introducing such structures would allow the categorical framework to capture forms of computational parallelism that are not addressed by the purely sequential model considered in the present work.

Another possible direction concerns enriched categorical structures incorporating quantitative invariants such as the structural metrics introduced in [3].

Further research may also explore algorithmic aspects of atomic normalization, including the complexity of constructing atomic representations and the computational behavior of structural reduction rules.

These directions, however, go beyond the structural scope of the present work.

8.5 Final Remarks

The categorical interpretation presented in this paper completes the structural program initiated in the preceding works of this series.

Atomic forms emerge as the canonical structural objects representing deterministic computations defined by straight-line programs. The normalization procedure provides a systematic way of passing from program-level descriptions to these canonical structures.

Within the categorical framework, atomic forms therefore serve as a fundamental representation layer for the study of structural properties of deterministic transitions.

This perspective opens the possibility of further connections between computational models, algebraic structures, and categorical methods in the study of deterministic computation.

9. Final Structural Perspective

The categorical framework developed in this paper completes the structural investigation of atomic representations of deterministic transitions initiated in the previous works of this series.

In order to summarize the resulting picture, we briefly describe the structural position of atomic forms within the broader theory of deterministic computation.

9.1 The Atomic Layer of Deterministic Computation

The results established across the four papers show that deterministic transitions definable by straight-line programs admit a natural structural representation in terms of atomic forms.
An atomic form is a finite directed acyclic structure whose vertices correspond to primitive operations from the signature Σ, whose edges represent data dependencies between operations, and whose ordered ports determine the precise ordering of arguments and interface connections.
The normalization procedure introduced in the first paper assigns to every SLP transition such a structure, producing an atomic representation that captures the computational dependencies of the transition in a canonical way.
From the perspective developed in this paper, atomic forms therefore constitute a structural layer that lies between program-level descriptions and the underlying semantic functions they compute.
This intermediate layer isolates the structural properties of computations independently of their syntactic realization as programs.

9.2 Structural Organization of the Atomic Space

The second paper of this series established that the space of atomic forms carries a natural algebraic structure.
When the interface type $X^n \to X^n$ is fixed, atomic forms admit a sequential composition defined by gluing the output interface of one form to the input interface of another. This composition operation is strictly associative and admits an identity atomic form, giving rise to a monoid structure on the space of atomic forms modulo isomorphism.
In addition, structural embeddings between atomic forms define a partial order on this space, capturing the idea that one atomic structure can occur as a substructure of another.
Local reduction rules further allow the simplification of atomic forms while preserving their semantics and respecting this structural order.
Together these constructions organize the space of atomic forms as a structured system of computational objects.

9.3 Quantitative Structure

The third paper introduced compositional structural metrics on atomic forms.
These metrics assign numerical values to atomic structures while respecting the algebraic and order-theoretic structure of the space. Examples include measures of computational depth, structural size, and potential parallelism.
The metrics are invariant under isomorphism, monotone with respect to structural embeddings, and subadditive with respect to composition.
Although these metrics provide a quantitative analysis of atomic structures, they are defined entirely at the atomic level and do not depend on the specific syntactic form of the original SLP programs.

9.4 Categorical Interpretation

The present paper shows that the normalization procedure has a precise categorical interpretation.
More specifically, we considered the category of SLP-definable deterministic transitions together with structural embeddings between them. We also considered the category of atomic forms modulo interface-preserving isomorphism.

Within this framework the normalization construction defines a functor

$$Norm: \boldsymbol{SLP}_{\Sigma} \to \boldsymbol{Atom}_{\Sigma},$$

,
and this functor forms a reflection with respect to the inclusion of atomic forms into SLP transitions.
This reflection theorem explains why atomic forms behave as canonical structural representatives of deterministic transitions: every SLP transition factors uniquely through its normalized atomic form with respect to morphisms into atomic structures.
Thus the categorical framework provides a conceptual foundation for the algebraic and metric structures previously introduced.

9.5 The Completed Structural Picture

Taken together, the four papers establish a coherent structural theory of atomic representations of deterministic transitions.
The overall structure of the theory can be summarized as follows.

1. Deterministic transitions definable by straight-line programs admit canonical atomic representations.
2. The space of atomic forms carries natural algebraic and order-theoretic structures.
3. Quantitative properties of atomic structures can be analyzed using compositional structural metrics.
4. The normalization procedure admits a categorical characterization as a reflection between the categories of SLP transitions and atomic forms.

Within this framework, atomic forms serve as the canonical structural objects representing deterministic computations.

9.6 Final Remarks

The purpose of the present series was to establish a rigorous structural foundation for atomic representations of deterministic transitions. The results obtained here show that atomic forms provide a natural level of abstraction for studying the structural properties of deterministic computations.
By separating structural dependencies from program syntax and from purely semantic descriptions of functions, atomic forms make it possible to analyze deterministic computations in terms of their intrinsic computational structure.
The categorical interpretation developed in this paper completes this program by clarifying the universal role played by atomic normalization within the theory.
This perspective suggests that atomic structures may serve as a useful foundation for further investigations into the structural theory of computation.

10. Acknowledgments and Final Notes

This section provides final acknowledgments and remarks concerning the preparation of the present work and its relation to the broader research program developed in this series of papers.

10.1 Acknowledgments

The author would like to acknowledge the role of constructive feedback and critical discussion in shaping the present work. The development of the structural framework presented in this paper

benefited from extensive internal review and stress-testing of definitions, proofs, and conceptual assumptions.
Particular attention was devoted to ensuring that the formal constructions introduced in the paper are mathematically precise and structurally consistent with the results established in the preceding works of the series.
The author also appreciates the broader mathematical literature that has developed the foundational tools used in this paper, including algebraic approaches to computation, graph-based models of programs, and categorical methods in the study of computational structures.

10.2 Relation to the Preceding Papers

The present paper forms the fourth part of a series devoted to the structural study of atomic representations of deterministic transitions.
The results of the series can be summarized briefly as follows.
The first paper established the existence of atomic normal forms for deterministic transitions definable by straight-line programs. It showed that every such transition can be represented by a directed acyclic computational structure composed of primitive operations from a fixed signature.
The second paper investigated the internal structure of the space of atomic forms. It introduced algebraic composition of atomic forms, structural embeddings, and reduction rules that preserve the semantics of the represented computation.
The third paper introduced compositional structural metrics on atomic forms, allowing quantitative analysis of computational structures while respecting the algebraic and order-theoretic properties of the atomic space.
The present paper provides a categorical interpretation of the normalization procedure introduced in the first work. It shows that atomic normalization defines a reflection between the category of SLP-definable deterministic transitions and the category of atomic forms.
Together these four works establish a unified structural theory of atomic representations for deterministic computations.

10.3 Availability of Definitions and Constructions

All definitions, constructions, and proofs presented in this paper are formulated in a fully explicit and constructive manner.
The atomic structures considered in the theory are finite directed acyclic graphs equipped with ordered ports and ordered interfaces. All structural operations—including composition, embeddings, and reductions—are defined directly in terms of these finite combinatorial objects.
As a consequence, the theoretical framework developed in this paper does not depend on external computational models or hidden semantic assumptions. Every object and transformation used in the theory can be described explicitly in terms of finite graph structures.
This constructive character is essential for the internal coherence of the structural theory developed across the series.

10.4 Final Perspective

The study of deterministic computation often focuses either on syntactic descriptions of programs or on purely semantic descriptions of the functions they compute.

The theory developed in this series instead emphasizes the **structural level** of computation: the level at which computations are represented as explicit dependency structures built from primitive operations.
Atomic forms provide a natural representation of this structural layer.
The categorical interpretation developed in the present work clarifies the mathematical role of atomic normalization within this framework and explains why atomic forms behave as canonical representatives of deterministic transitions.
This structural perspective suggests that atomic representations may serve as a useful foundation for further investigations into the structural and algebraic properties of computational processes.

11. Concluding Remarks

The present work completes the structural investigation of atomic representations of deterministic transitions initiated in the preceding papers of this series. The results obtained across these works establish a coherent framework for studying deterministic computations through their underlying structural representations.
The central idea of the theory is that deterministic transitions definable by straight-line programs admit canonical representations in the form of finite directed acyclic structures composed of primitive operations from a fixed signature. These structures, referred to as **atomic forms**, provide a natural level of abstraction at which the structural properties of computations can be analyzed independently of specific program syntax.
The normalization procedure introduced in the first paper of the series provides a systematic method for constructing atomic representations of deterministic transitions. This procedure transforms program-level descriptions into explicit dependency structures that make the computational organization of the transition visible.
The second paper investigated the internal structure of the space of atomic forms. It introduced a sequential composition operation obtained by gluing ordered interfaces, and it showed that the resulting structure forms a monoid when the interface type is fixed. Structural embeddings between atomic forms were also introduced, inducing a partial order on the space of atomic structures.
The third paper extended the theory by introducing compositional structural metrics that measure quantitative properties of atomic forms. These metrics are compatible with the algebraic and order-theoretic structure of the atomic space and allow a systematic quantitative analysis of computational structures.
The present paper provides a categorical interpretation of the normalization process and shows that atomic normalization admits a natural universal characterization. More precisely, the normalization construction defines a reflection between the category of SLP-definable deterministic transitions and the category of atomic forms modulo interface-preserving isomorphism.
This categorical perspective clarifies the structural role of atomic forms within the theory. It shows that atomic forms act as canonical representatives of deterministic transitions with respect to structural embeddings into atomic structures.
Taken together, the results of the series provide a unified structural framework for deterministic transitions based on three complementary components:

- a normalization procedure that produces atomic representations,
- algebraic and order-theoretic structures on the space of atomic forms,
- and quantitative metrics that measure structural properties of these forms.

Within this framework, atomic forms emerge as the natural structural objects for representing deterministic computations.
Although the present work focuses specifically on transitions definable by straight-line programs, the structural viewpoint developed here suggests possible extensions of the theory. In particular, future work may investigate the role of atomic structures in more general computational models, the introduction of parallel composition structures, or the interaction between categorical structure and quantitative metrics.
The theory developed in this series therefore provides a structural foundation for further study of computational representations based on explicit dependency structures.
The notion of atomic normalization and the associated structural framework introduced in this series provide a new structural perspective on deterministic computation based on canonical port-structured dependency graphs.

12. Related Work and Structural Distinctions

The theory developed in this series of papers is related to several well-established areas of research, including graph representations of computations, algebraic models of programs, and categorical approaches to computation. In this section we briefly clarify how the present framework relates to these existing lines of work.
The goal of this discussion is not to provide an exhaustive survey of the literature, but rather to position the structural theory of atomic forms within the broader landscape of computational models.

Novelty of the Categorical Construction.

To the best of the author's knowledge, the categorical construction developed in this work has not previously been formulated in the literature.
In particular, the following elements appear to be new:

- the definition of atomic forms as port-structured dependency graphs with ordered interfaces,
- the canonical normalization of SLP-definable deterministic transitions into atomic forms,
- the construction of a normalization functor linking SLP transitions and atomic forms,
- the characterization of this normalization as a categorical reflection.

No previous work appears to establish a reflective subcategory connecting straight-line program transitions and port-structured atomic dependency graphs in this manner.
While atomic forms share similarities with circuit and dataflow representations, the present framework differs by introducing ordered vertex ports, canonical normalization of SLP transitions, and a categorical reflection structure that does not appear in classical DAG-based computational models.

12.1 Graph Representations of Computations

Many models of computation represent programs or computations as graphs describing dependencies between operations.
Examples include:

- arithmetic circuits,
- dataflow graphs,
- program dependence graphs,
- and various graph-based representations used in compiler theory.

In such models, vertices typically represent operations and edges represent the flow of values between operations. These representations make it possible to analyze structural properties of

computations, such as dependency structure, evaluation order, and opportunities for parallel execution.
Atomic forms belong to this general family of graph-based computational representations. Like arithmetic circuits and dataflow graphs, they represent computations as directed acyclic graphs whose nodes correspond to primitive operations.
However, atomic forms differ from many existing graph models in several important respects. First, atomic forms explicitly incorporate **ordered input ports** for every vertex. This ordering determines the argument structure of the corresponding operation and forms an explicit part of the structural representation.
Second, atomic forms include **ordered interface ports** that describe the external boundary of the computation. These ports provide a precise mechanism for composing atomic forms by gluing interfaces.
Third, the theory developed in this series systematically studies the algebraic and categorical structure of the space of such graphs.

12.2 Straight-Line Programs

Straight-line programs (SLPs) are a classical model of deterministic computation in which a computation is described as a sequence of assignments without branching or loops.
SLPs are widely used in several areas, including

- algebraic complexity theory,
- symbolic computation,
- and the study of arithmetic circuits.

The normalization procedure introduced in the first paper of this series establishes a structural connection between SLP computations and atomic forms. Specifically, every SLP-definable deterministic transition can be transformed into an atomic form representing its computational dependency structure.
This relationship allows SLP computations to be analyzed through the structural properties of their atomic representations.

12.3 Algebraic Models of Computation

Several algebraic frameworks have been developed to study computations through algebraic operations on programs or computational structures.
Examples include

- semiring-based models of computation,
- algebraic theories of programs,
- and algebraic approaches to circuit complexity.

The algebraic structure introduced in the second paper of this series contributes to this line of work by showing that atomic forms admit a natural composition operation that forms a monoid when the interface type is fixed.
This composition corresponds to sequential execution of computations and allows atomic forms to be treated as algebraic objects.
Unlike many algebraic program models, however, the present framework focuses specifically on **explicit structural representations** of computations rather than abstract algebraic expressions.

12.4 Categorical Approaches to Computation

Category theory has been widely used to study computational structures. Examples include

- categorical semantics of programming languages,
- monoidal categories for modeling computation,
- and categorical models of circuits and processes.

In the present work we use categorical language in a limited but precise way.
The main categorical result of the paper shows that atomic normalization defines a **reflection** between the category of SLP transitions and the category of atomic forms.
This result provides a conceptual explanation for the canonical role of atomic forms within the structural theory developed in this series.
At the same time, the categorical framework used here remains intentionally minimal. In particular, the present work does not introduce tensor structures, enriched categories, or higher categorical constructions.

12.5 Position of the Present Theory

The framework developed in this series occupies a specific position within the broader landscape of computational models.
The theory focuses on deterministic computations that

- are definable by straight-line programs,
- admit explicit dependency structures,
- and can be represented as finite directed acyclic graphs over a fixed signature.

Within this domain, atomic forms provide a canonical structural representation that separates the structural organization of a computation from its syntactic program description.
The algebraic, order-theoretic, metric, and categorical structures developed across the four papers together form a coherent framework for studying these representations.
This structural viewpoint complements existing approaches to computation and provides a foundation for further investigations into the mathematical structure of deterministic computational processes.

12.6 Distinction from Related Graph and Categorical Frameworks

The framework developed in this series is related to several existing models of computation and graph-based representations. However, it differs from these models in several essential structural aspects.

PROPs and operads.
PROPs and operads describe algebraic structures generated by operations with multiple inputs and outputs. Such structures typically assume symmetric monoidal composition and tensor products. The framework developed here does not introduce tensor structure or symmetry operations. Instead, it focuses on sequential composition of port-structured dependency graphs. Consequently, the category of atomic forms is not formulated as a PROP or operad.

String diagrams and traced monoidal categories.
String diagram frameworks represent morphisms in monoidal categories where wires may bend or form feedback loops via trace operators. Atomic forms, in contrast, are strictly acyclic directed structures with ordered ports and interfaces. No trace operation or wire bending is introduced in the present theory.

Term graphs.
Term graphs are directed acyclic graphs representing shared subterms in algebraic expressions. While atomic forms share the DAG property, they differ by incorporating explicit ordered vertex ports and ordered input/output interfaces that serve as structural boundaries for composition.

Circuit and dataflow categories.
Graph-based representations of circuits and dataflow computations also use directed acyclic graphs of operations. However, these frameworks typically do not include canonical normalization procedures producing unique representatives up to interface-preserving isomorphism. The normalization construction developed in this series therefore introduces an additional structural layer absent from classical circuit models.
Compiler intermediate representations (SSA / IR lowering).
Compiler representations such as SSA or lowering transformations operate on syntactic program structures and depend on compilation strategies. In contrast, atomic normalization provides a canonical structural representation characterized by a universal categorical property. It does not depend on program syntax or compilation procedures.

Graph rewriting systems.
Graph rewriting frameworks study transformations of graphs via local rewrite rules. The present work does not introduce rewriting systems or rewriting semantics. Instead, it focuses on canonical normalization and structural embeddings between atomic dependency graphs.
While atomic forms share certain similarities with directed acyclic computational graphs, the present framework introduces several structural features not present in standard DAG models. In particular, atomic forms incorporate ordered vertex input ports, explicit ordered interfaces, and a canonical normalization procedure characterized by a categorical reflection property. These elements distinguish the theory from classical graph-based program representations and from compiler intermediate representations such as SSA.

12.7 Distinction of Structural Embeddings
Structural morphisms used in this framework differ from classical subgraph embeddings.
A structural morphism between atomic forms is defined as an injective mapping of vertices that preserves
• operation labels,
• ordering of input ports,
• attachment of edges to specific ports,
• ordered input and output interfaces.
In contrast, classical subgraph embeddings usually preserve only vertices and edges of a graph. The additional port- and interface-preserving constraints ensure that structural morphisms respect the computational structure of atomic forms.
These additional constraints play a crucial role in the categorical construction developed in this work.

References

1. (1) A. Zubov,
   Atomic Normalization of SLP-Definable Deterministic Transitions, Preprint, 2026..
2. (2) A. Zubov,
   Algebraic Structure of Atomic Forms over a Fixed Signature, Preprint, 2026.
3. (3) A. Zubov,
   Compositional Structural Metrics for Atomic Computational Forms, Preprint, 2026.
4. (4) J. E. Hopcroft and J. D. Ullman,
   Introduction to Automata Theory, Languages, and Computation, Addison-Wesley, 1979.

5. (5) A. V. Aho, J. E. Hopcroft and J. D. Ullman,
   The Design and Analysis of Computer Algorithms, Addison-Wesley, 1974.
6. (6) L. G. Valiant, Completeness Classes in Algebra,
   Proceedings of the 11th Annual ACM Symposium on Theory of Computing (STOC), 1979.
7. (7) I. Wegener, The Complexity of Boolean Functions,
   Wiley-Teubner, 1987.
8. (8) J. E. Savage, Models of Computation: Exploring the Power of Computing,
   Addison-Wesley, 1998.
9. (9) S. Mac Lane, Categories for the Working Mathematician,
   Springer, 1971.
10. (10) S. Awodey, Category Theory,
    Oxford University Press, 2010.
11. (11) D. I. Spivak, Category Theory for the Sciences,
    MIT Press, 2014.
12. (12) S. Arora and B. Barak, Computational Complexity: A Modern Approach,
    Cambridge University Press, 2009.