SSCCS

SSCCS Foundation, , contact@ssccs.org

What is SSCCS

SSCCS (Schema—Segment Composition Computing System) is a definition of what computation is, not a prescription for how to build it. The name itself encodes the ambition: a single abstraction over all coordinate-constrained domains. Spec Space (instruction encoding, configuration enumeration) and State Space (knowledge indexing, storage, retrieval) are two prominent instances—but the framework is not limited to either. Any domain that can be described with coordinates, constraints, and observation is a subspace of the SSCCS model. This includes, but is not limited to, instruction encoding spaces, knowledge organization spaces, hardware topology spaces, configuration spaces, and design spaces. There is no ceiling. Every new domain reduced to the four primitives expands the model without changing it.

It treats computation as **the observation of fixed structure under changing conditions.** Data does not move. Instructions do not execute sequentially. Instead, a static blueprint (Scheme) binds coordinate points (Segments), dynamic rules (Field) constrain what is admissible, and the active act of Observation collapses that constrained space into a result (Projection).

The four primitives form a complete computational model with formal definitions, a working Rust reference implementation, and a growing ecosystem of derived products. Each product applies the four primitives to a specific domain—verification, storage, hardware design, AI context management—without modifying the core model. The long-term trajectory envisions purpose-built hardware that instantiates the model natively. In the interim, empirical calibrations on existing silicon confirm the model is not merely theoretical.

Why This Matters Now

The von Neumann model—instructions operating on mutable memory, step by step—has driven computing for eighty years. It is reaching fundamental limits:

- Data movement dominates energy: 60–80% of energy in modern systems is spent moving data between memory and processor, not computing.
- Parallelism is bolted on: Every core, thread, and SIMD lane requires explicit programmer effort. Implicit parallelism does not exist.
- Verifiability is retrofitted: Determinism is not guaranteed by the architecture. Security, auditability, and correctness are added after the fact.

SSCCS addresses all three at the architectural level. Segments are immutable; any number of observations can proceed concurrently without locks or races. Data stays in place; only projections travel. Observation is deterministic by definition; replaying the same Scheme, Field, and time coordinate always yields the same projection.

The Four Primitives

Segment is an immutable coordinate point with a unique cryptographic ID. It stores no value—only its position exists. The actual value is revealed later through Observation.

- Immutable by construction.
- Stateless: holds only coordinates and identity.
- Analogous to a point on a map: no information, but a definite location.

Scheme is a structural blueprint that binds Segments into a topology. It defines axes (coordinate dimensions), adjacency (how Segments connect), memory layout (how Segments map to physical storage), and observation rules (how Projection is computed).

- Examples: line, 2D grid, 3D tensor, graph, custom topology.
- Schemes are immutable. The structure does not change during computation.
- The compiler analyzes the Scheme to produce an optimal physical memory layout.

Field is a dynamic constraint substrate. It imposes conditions on the Scheme: value bindings, arithmetic operations, comparison thresholds, composition rules. Fields are the only mutable layer.

- Fields can be updated at runtime.
- Multiple Fields compose via intersection (AND), union (OR), or sequential chaining.
- Every Field update is versioned. Staleness is tracked at the Field level.

Observation is the only active event. It takes a Scheme and a Field, evaluates the constrained structure, and produces a Projection. Observation is a pure function: the same Scheme, Field, and time coordinate always produce the same Projection. This makes every observation independently verifiable and replayable.

- Deterministic: same inputs always yield the same output.

- Stateless: the result is transient. To retain it, the caller records it externally (neXus stores Observations as immutable Facts).
- Implicitly parallel: because Segments are immutable, observations on disjoint regions proceed concurrently with zero programmer effort.

The Ecosystem

SSCCS is not a single product. It is the conceptual foundation for a growing family of projects across different domains, each applying the four primitives to a specific space. The Project Ecosystem lists all active projects. Each is independently deployable; together they form a distributed Scheme where every Observation in one domain enriches the whole.

Relationship with Conventional Computing

SSCCS is not a replacement for von Neumann computing, nor does it subsume it. It is a different kind of thing altogether: a definitional framework rather than a specific computational theory. Von Neumann computing, dataflow, quantum computing, neural computation—each can be described within the SSCCS coordinate system, but none is required to be. The relationship is not hierarchical containment. It is referential: any computational model can be expressed in terms of the four primitives if one chooses to, but the model's validity does not depend on being expressed that way. SSCCS defines a language for describing computation, not a particular computation to be performed.

Verifiability is structural. Every Observation is deterministic and traceable from blueprint to result. There is no runtime-dependent behavior to verify after the fact. The audit log is the computation itself. Parallelism is implicit. Immutable Segments cannot race. Observations on disjoint regions of a Scheme proceed concurrently without locks, without programmer effort, and without correctness risk. Data movement is eliminated. Projections travel, not data. Projections travel, not data. The result of an Observation is a projection—a compact representation of the constrained structure, not a relocated copy of the underlying Segments.

Stigmergic Evolution

SSCCS treats its own development as an instance of the model. The ecosystem forms a distributed Scheme across problem-space. Each project is a Segment with its own coordinate; Field constraints (market conditions, hardware capabilities, research findings) evolve continuously; Observations (releases, experiments, deployments) produce projections that feed back into the Field. The system evolves like a biological ecosystem: each component adapts to its local environment while the whole maintains structural coherence through the shared coordinate system.

This dynamic—implicit coordination through a shared substrate rather than central planning—enables extreme parallelism in development. Multiple agents observe different regions of the problem-space concurrently. Because Segments (projects, specifications, implementations) are immutable, there is no merge conflict. Only projections travel between components. Development velocity scales with the number of independent observers, not with coordination overhead. The neXus project, for instance, applies this exact coordination layer to multi-agent knowledge storage and retrieval—an apparently unrelated domain that nevertheless collapses to the same four primitives.

- Whitepaper: HTML / PDF — DOI: 10.5281/zenodo.18759106 via CERN/Zenodo, indexed by OpenAIRE. Licensed under *CC BY-NC-ND 4.0*.
- Official repository: GitHub. Licensed under *Apache 2.0*.
- Governed by the Foundational Charter and Statute of the SSCCS Foundation (in formation).
- Provenance: Human-in-Command, AI-assisted. Aligns with ISO/IEC JTC 1/SC 42 and C2PA-certified. Full intellectual responsibility with author(s).