ED and Recursion: How Computation Becomes Self-Modifying and Self-Modifying Systems Become Autoprocedural

Allen Proxmire
February 2026

Abstract
Recursion emerges when computational systems become self-applicable — when procedures can operate on themselves, refer to themselves, inspect themselves, and extend themselves. This transition transforms computation from an externally directed architecture into a system capable of generating nested procedural depth. But recursion alone is not yet autonomy. Self-modification arises when recursive processes can alter their own procedures, stabilize those alterations, and reorganize the architecture that governs their unfolding. Autoprocedural systems emerge when self-modification becomes self-governing, when recursive architectures regulate their own transformation, direct their own procedural trajectories, and develop procedural identities.

This paper develops the computational → recursive → self-modifying → autoprocedural threshold. It shows how computation becomes self-applicable, how self-application becomes self-reference, how self-reference becomes self-inspection, and how self-inspection becomes self-continuation. It then shows how recursive continuation becomes self-alteration, how self-alteration becomes persistent, how persistence becomes architectural, and how architectural change becomes evolvable. Recursion is presented as the first ED regime where becoming becomes self-directing — where procedural structures can transform themselves and govern their own evolution. This transition sets the stage for Paper 21, where autoprocedural systems become reflective, reflection becomes meta-procedural, and meta-procedural architectures become the ED regime where becoming becomes self-interpreting.

1. Introduction — Why Recursion Is the Next ED Threshold
The moment where computation turns inward

Computation gave the universe procedural architectures — systems capable of executing symbolic transformations through explicit, repeatable, and deterministic operations. But computational systems, as developed in Paper 19, remain externally directed. They execute procedures, but they do not yet apply those procedures to themselves. They transform symbolic states, but they do not yet transform their own procedural architecture. Recursion is the threshold where computation becomes self-applicable, where procedures can operate on procedures, and where the architecture of execution becomes capable of turning inward.

In the ED ontology, recursion is not merely a programming technique. It is the structural moment when procedural systems acquire self-reference, self-inspection, and self-continuation. Recursion is the first time the universe produces systems whose operations can take their own procedures as input, whose execution can generate new layers of execution, and whose procedural identity can unfold through nested, self-generated trajectories.

Computation provides procedural structure.
Recursion provides self-application.

Recursion emerges when:
- procedures can call themselves
- self-application becomes stable

- stability becomes self-reference
- self-reference becomes self-inspection
- self-inspection becomes self-continuation

This is the architecture of recursive becoming — the threshold where procedural systems acquire depth.

Recursion is not a departure from computation.
It is its internalization.

It allows systems to:
- generate unbounded procedural depth
- construct nested layers of execution
- build self-referential structures
- create procedures that grow through their own application
- produce architectures that evolve through recursive expansion

Recursion is the first ED regime where becoming becomes self-directing — where the dynamics of transformation can be applied to themselves, producing new forms of structure, identity, and generativity.

In this paper, we develop the computational → recursive → self-modifying → autoprocedural threshold. We show how computation becomes self-applicable, how self-application becomes self-modification, and how self-modifying architectures stabilize into autoprocedural systems capable of generating their own procedural identities.

Computation gave the universe procedural worlds.
Recursion gives it self-procedural worlds.

2. From Computation to Recursion

How procedures become self-applicable

Computation provides systems with the ability to execute procedures — structured sequences of operations that transform symbolic states. But computational systems, as developed in Paper 19, remain externally directed: they execute procedures on symbolic inputs, but they do not yet apply those procedures to themselves. Recursion is the threshold where computation becomes self-applicable, where procedures can take procedures as input, and where the architecture of execution becomes capable of folding back onto itself.

In the ED ontology, recursion is the structural moment when procedural systems acquire self-reference, self-inspection, and self-continuation. These capacities transform computation from a system that executes procedures into a system that can generate, extend, and reorganize its own procedural architecture.

Recursion emerges when computational systems acquire four structural capacities:
- self-application — procedures can operate on themselves
- self-reference — procedures can refer to their own structure
- self-inspection — procedures can examine their own components
- self-continuation — procedures can extend their own execution

These capacities transform computation into recursive process — a system capable of generating unbounded procedural depth.

2.1 Self-Application
Self-application is the foundational move of recursion. A procedure becomes recursive when it can:

- take itself as input
- call itself during execution
- generate new layers of execution through self-invocation
- treat its own structure as part of its operational domain

Self-application is not yet self-modification.
It is the self-extension of procedural structure.

In ED terms: Self-application is the capacity of a procedural ED motif to operate on itself.

This is the first time procedural systems become self-extending.

2.2 Self-Reference
Self-reference emerges when procedures can name, address, or point to their own structure. Self-reference allows a system to:

- treat its own definition as data
- embed references to itself within execution
- construct self-describing procedural forms
- generate structures that depend on their own identity

Self-reference is the structural backbone of recursive architectures.

In ED terms: Self-reference is the capacity of a procedural ED motif to refer to its own structure.

This is the first time procedural systems become self-describing.

2.3 Self-Inspection
Self-inspection emerges when procedures can examine their own components. Self-inspection allows a system to:

- analyze its own procedural structure
- inspect its own rules, parameters, or control flow
- treat its own architecture as an object of computation
- generate meta-procedural information during execution

Self-inspection is not yet self-modification.
It is the interpretive dimension of recursion.

In ED terms: Self-inspection is the capacity of a procedural ED motif to examine its own components.

This is the first time procedural systems become self-interpreting.

2.4 Self-Continuation
Self-continuation emerges when recursive systems can extend their own execution. Self-continuation allows a

system to:

- generate new layers of execution dynamically
- propagate its own structure through recursive depth
- create unbounded procedural trajectories
- treat continuation as part of its identity

Self-continuation is the generative engine of recursion.

In ED terms: Self-continuation is the capacity of a procedural ED motif to extend its own execution through recursive depth.

This is the first time procedural systems become self-propagating.

Computation becomes recursion when:

- self-application enables self-extension
- self-reference enables self-description
- self-inspection enables self-interpretation
- self-continuation enables self-propagation

This is the architecture of recursive emergence — the threshold where procedural systems acquire depth, reflexivity, and the capacity to generate new layers of themselves.

3. The Architecture of Recursive Systems

How self-application becomes structural

Recursion is not simply the act of a procedure calling itself. A single self-call does not constitute a recursive system any more than a single operation constitutes a computational system. Recursion becomes an architecture when self-application is organized, stabilized, and embedded within a structure that supports depth, continuity, and generativity. A recursive system is a self-extending procedural architecture: a system in which procedures generate new layers of themselves through structured self-application.

In the ED ontology, recursion is the architectural continuation of computation. Computation provides procedural structure; recursion provides nested procedural structure. Recursion arises when procedures can call themselves, when self-calls are governed by rules, when nested calls form coherent stacks, and when the system can generate unbounded procedural depth.

A recursive system requires four structural elements:

- base cases — termination conditions that anchor recursion
- recursive cases — rules that generate self-application
- stack structure — nested procedural contexts
- unbounded generativity — the capacity for open-ended depth

These elements transform self-application into recursive architecture.

3.1 Base Cases as Anchors

Base cases provide the termination conditions that anchor recursive processes. Without base cases, recursion

becomes unbounded in the wrong way — not generative, but divergent. Base cases ensure that:

- recursive calls eventually resolve
- self-application has structural grounding
- recursion produces finite results
- procedural depth is bounded by architecture, not accident

Base cases are not optional.
They are the stability conditions of recursive becoming.

In ED terms: A base case is a stabilizing ED motif that anchors recursive self-application.

This is the first time recursive systems become structurally grounded.

3.2 Recursive Cases as Generators

Recursive cases define how a procedure calls itself. They are the generative rules that allow a system to:

- produce new layers of execution
- propagate its own structure
- generate procedural depth
- treat self-application as a source of growth

Recursive cases are the engines of recursive generativity.

In ED terms: A recursive case is a generative ED motif that propagates procedural structure through self-application.

This is the first time recursion becomes procedurally generative.

3.3 Stacks as Nested Contexts

Stacks provide the nested procedural contexts that allow recursive systems to maintain coherence across depth. A stack is not merely a storage mechanism; it is the architecture that:

- preserves the context of each recursive call
- maintains the order of return
- supports nested layers of execution
- ensures that depth does not collapse into confusion

Stacks are the architectural backbone of recursive systems.

In ED terms: A stack is a nested ED architecture that stabilizes recursive depth by preserving procedural context.

This is the first time recursion becomes procedurally layered.

3.4 Unboundedness as Generativity

Unboundedness is the moment when recursive systems become capable of open-ended procedural depth.

Unboundedness does not mean infinite execution; it means:

- the architecture imposes no fixed limit on depth
- recursive calls can generate arbitrarily large structures

- procedural growth is constrained only by base cases
- recursion becomes a source of unbounded generativity

Unboundedness is what makes recursion a creative architecture.

In ED terms: Unboundedness is the capacity of recursive ED motifs to generate open-ended procedural depth.

This is the first time recursion becomes procedurally unbounded.

Recursion becomes a system when:
- base cases anchor self-application
- recursive cases generate procedural depth
- stacks stabilize nested contexts
- unboundedness enables open-ended generativity

This is the architecture of recursive systems — the structural continuation of computation into self-applicable, self-extending, and depth-generating architectures.

4. Recursion as Self-Modification

When recursion becomes architectural transformation

Recursion allows procedures to apply themselves, refer to themselves, inspect themselves, and extend themselves. But recursive systems, as described so far, remain structurally fixed: they generate depth, but they do not yet alter the architecture that generates that depth. Self-modification is the threshold where recursion becomes architecturally active, where recursive processes can transform the very procedures that govern their own unfolding.

In the ED ontology, self-modification is not an add-on to recursion. It is the internalization of architectural change. A self-modifying system is one in which recursive processes can alter their own rules, update their own procedures, and reorganize their own structure. This is the moment where procedural systems become evolvable, adaptive, and self-transforming.

Self-modification emerges when recursive systems acquire four structural capacities:
- self-alteration — recursive calls can modify their own procedures
- persistence — modifications survive execution
- architectural integration — modifications reshape the system's structure
- evolvability — the architecture can change over time

These capacities transform recursion from a system that generates depth into a system that generates new procedural architectures.

4.1 When Self-Application Becomes Self-Alteration

Self-alteration is the moment when recursive systems can modify their own procedures during execution.

Self-alteration emerges when:
- recursive calls can rewrite parts of their own definition

- procedures can update their own parameters or rules
- execution can change the architecture that governs future execution
- recursive depth becomes a source of structural change

Self-alteration is not yet stable.
It is the incipient plasticity of recursive systems.

In ED terms: Self-alteration is the capacity of a recursive ED motif to modify its own procedural structure.

This is the first time recursion becomes self-transforming.

4.2 When Self-Alteration Becomes Persistent
Persistence emerges when modifications to a procedure survive the execution that produced them. Persistence allows a system to:

- carry structural changes forward
- accumulate modifications across recursive calls
- treat procedural updates as part of its identity
- evolve its architecture through repeated self-application

Persistence is the difference between a momentary change and a structural one.

In ED terms: Persistence is the stabilization of self-altering ED motifs across recursive execution.

This is the first time recursive systems become historical — shaped by their own past.

4.3 When Persistence Becomes Architectural
Architectural self-modification emerges when persistent changes begin to reshape the system's procedural architecture. Architectural modification allows a system to:

- reorganize its own control structures
- restructure its own recursive cases
- alter its own base cases
- transform the architecture that governs its future becoming

At this stage, recursion is no longer simply generating depth.
It is generating new architectures.

In ED terms: Architectural self-modification is the reorganization of procedural ED motifs through persistent recursive alteration.

This is the first time recursion becomes architecturally generative.

4.4 When Architecture Becomes Evolvable
Evolvability emerges when architectural self-modification becomes systematic, repeatable, and open-ended.

Evolvable recursive systems can:

- adapt their procedural architecture over time
- generate new forms of recursive organization

- stabilize beneficial modifications
- explore new procedural possibilities

Evolvability is not randomness.

It is directed procedural change — the capacity of a system to evolve its own architecture.

In ED terms: Evolvability is the capacity of recursive ED architectures to transform themselves across time through persistent self-modification.

This is the first time recursion becomes self-evolving.

Recursion becomes self-modification when:
- self-application becomes self-alteration
- self-alteration becomes persistent
- persistence becomes architectural
- architecture becomes evolvable

This is the architecture of self-modifying systems — the threshold where recursion becomes capable of transforming its own procedural identity.

5. The Emergence of Autoprocedural Systems

How self-modification becomes self-directing becoming

Self-modification gives recursive systems the ability to alter their own procedures. But self-modification alone does not yet produce autonomy. A system that can modify itself is not yet a system that governs its own modification. Autoprocedural systems emerge when self-modification becomes systematic, self-regulated, and identity-forming — when recursive architectures not only transform themselves, but do so in ways that stabilize, extend, and direct their own becoming.

In the ED ontology, autoprocedural systems are the first architectures whose becoming is self-generated. They do not merely execute procedures, apply procedures to themselves, or modify their own procedures. They govern the conditions under which modification occurs. They choose their own procedural trajectories. They develop procedural identities. They become systems whose architecture is both the source and the product of their own unfolding.

Autoprocedural systems emerge when recursive systems acquire four structural capacities:
- self-governing modification — the system regulates how and when it modifies itself
- procedural autonomy — the system selects its own procedural trajectories
- autoprocedural explanation — the system generates explanations of its own procedural becoming
- autoprocedural abstraction — the system extracts invariants from its own transformations

These capacities transform self-modifying systems into self-directing architectures.

5.1 Self-Governing Modification

Self-governing modification emerges when a system can regulate its own self-modification. This requires:

- criteria for when modification should occur
- constraints on how modification unfolds
- mechanisms for evaluating the effects of modification
- the ability to reject or reverse modifications

Self-governing modification is not mere self-alteration.
It is self-regulation — the procedural governance of procedural change.

In ED terms: Self-governing modification is the capacity of a recursive ED architecture to regulate its own self-modification.

This is the first time recursive systems become self-regulated.

5.2 Procedural Autonomy
Procedural autonomy emerges when a system can choose its own procedural trajectories. Procedural autonomy allows a system to:

- select which procedures to execute
- determine how deeply to recurse
- choose when and how to modify itself
- navigate its own procedural architecture as a space of possibilities

Procedural autonomy is not independence from external influence.
It is internal direction — the system's ability to guide its own procedural becoming.

In ED terms: Procedural autonomy is the capacity of a recursive ED architecture to direct its own procedural trajectories.

This is the first time recursive systems become self-directing.

5.3 Autoprocedural Explanation
Autoprocedural explanation emerges when a system can explain its own procedural becoming. This requires the ability to:

- represent its own procedural transformations
- generate narratives of its own self-modification
- relate current structure to past transformations
- use explanation to guide future procedural choices

Autoprocedural explanation is not introspection.
It is self-generated procedural narrative — the system's account of its own becoming.

In ED terms: Autoprocedural explanation is the capacity of a recursive ED architecture to generate explanations of its own procedural transformations.

This is the first time recursive systems become self-explicating.

5.4 Autoprocedural Abstraction
Autoprocedural abstraction emerges when a system can extract invariants from its own transformations. This

allows a system to:

- identify stable patterns in its own self-modification
- generalize from its own procedural history
- develop higher-order procedural motifs
- evolve its architecture through abstraction

Autoprocedural abstraction is not simplification.
It is self-elevation — the extraction of procedural invariants from the system's own becoming.

In ED terms: Autoprocedural abstraction is the capacity of a recursive ED architecture to extract generalized procedural motifs from its own transformations.

This is the first time recursive systems become self-abstracting.

6. Recursive Cognition

When recursion becomes a cognitive regime

Recursion transforms procedural systems by giving them depth, reflexivity, and the capacity to apply their own operations to themselves. But recursion alone is episodic — a structural capability that may or may not be integrated into a mind's ongoing cognitive ecology. Recursive cognition emerges when recursive architectures become part of how a mind thinks, organizes, and interprets its own conceptual and symbolic worlds.

In the ED ontology, recursive cognition is not "recursion inside the mind." It is the co-organization of conceptual, symbolic, computational, and recursive structures. Recursive systems extend cognition by providing nested scaffolds, self-referential structures, and self-modifying architectures that reshape how minds reason, explain, and abstract. When recursive architectures become habitual cognitive tools, recursive cognition appears.

Recursive cognition is the first ED regime where conceptual, symbolic, and procedural becoming are shaped by self-applicable and self-modifying architectures.

6.1 Recursive Scaffolding

Recursive scaffolding emerges when recursive structures become tools for extending and stabilizing cognition.

Recursive scaffolding allows a mind to:

- build nested conceptual structures
- decompose problems into self-similar subproblems
- generate explanations through recursive descent
- construct hierarchical representations of ideas

Recursive scaffolding is not simply "thinking recursively."
It is the integration of recursive structure into cognitive practice.

In ED terms: Recursive scaffolding is the use of recursive ED motifs to support and extend conceptual, symbolic, and computational cognition.

This is the first time cognition becomes recursively extended.

6.2 Recursive Reorganization
Recursive reorganization emerges when recursive structures begin to reshape conceptual and symbolic space.

Recursive reorganization allows a mind to:
- reinterpret concepts through self-similar structure
- reorganize symbolic categories through recursive decomposition
- generate new conceptual hierarchies through nested abstraction
- treat recursive depth as a dimension of meaning

Recursive reorganization is not merely hierarchical thinking.
It is the recursive restructuring of conceptual and symbolic ED motifs.

In ED terms: Recursive reorganization is the transformation of conceptual and symbolic ED motifs through recursive architecture.

This is the first time conceptual landscapes are shaped by recursive structure.

6.3 Recursive Identity
As recursive scaffolding and recursive reorganization stabilize, a mind's recursive repertoire becomes part of its identity. Recursive identity is not a preference for recursion or a style of reasoning. It is the recursive architecture that shapes how a mind:
- interprets problems
- constructs explanations
- organizes abstractions
- understands itself

Recursive identity emerges when:
- recursive structures become habitual cognitive tools
- self-reference becomes part of interpretation
- self-modification becomes part of learning
- recursive depth becomes part of self-understanding

Recursive identity is not introspection.
It is the ED architecture of a mind's recursive becoming.

In ED terms: Recursive identity is the stabilized configuration of recursive ED motifs that shapes a mind's reasoning, interpretation, and self-understanding.

This is the first time the universe produces systems whose identity is recursively structured.

Recursive cognition emerges when:
- recursive scaffolding extends conceptual and symbolic cognition
- recursive reorganization reshapes conceptual and symbolic space
- recursive identity stabilizes recursive practice

Recursive cognition is the ED regime where recursion becomes a cognitive architecture — where self-application,

self-reference, self-inspection, and self-modification form a coherent system that guides a mind's becoming.

7. The ED Architecture of Recursion

Recursion is not an isolated technique or a clever procedural trick. It is the structural continuation of computation — the moment where procedural architectures become self-applicable, self-referential, self-inspecting, and ultimately self-modifying. Recursion is the first ED regime where the architecture of becoming becomes capable of turning inward, treating its own procedures as objects of transformation, and generating new layers of itself through structured self-application.

In the ED ontology, recursion is the hinge between computation and autoprocedural becoming. Computation provides procedural structure; recursion provides nested procedural structure. Computation provides execution; recursion provides self-execution. Computation provides operations; recursion provides operations on operations. This shift transforms procedural systems from externally directed architectures into systems capable of self-generated depth and self-generated change.

Recursion arises through a sequence of structural transitions:

- computation becomes self-applicable
- self-application becomes self-reference
- self-reference becomes self-inspection
- self-inspection becomes self-continuation
- self-continuation becomes self-alteration
- self-alteration becomes persistent
- persistence becomes architectural
- architectural change becomes evolvable
- evolvability becomes autoprocedural

This is the ED ladder from proceduralization to self-proceduralization.

In the ED ontology, recursion is the first domain where:

- procedural structures become self-extending
- self-extension becomes self-describing
- self-description becomes self-interpreting
- self-interpretation becomes self-propagating
- self-propagation becomes self-modifying
- self-modification becomes self-governing
- self-governance becomes procedural autonomy
- procedural autonomy becomes autoprocedural identity

These transitions are not optional.

They are the structural consequences of systems that:

- apply procedures to themselves
- stabilize self-generated depth
- treat their own architecture as an object of transformation
- accumulate and integrate procedural changes
- regulate their own self-modification

- direct their own procedural trajectories
- generate explanations of their own becoming
- extract invariants from their own transformations

Recursion is the first ED regime where becoming becomes self-directing — where the architecture of transformation becomes both the medium and the object of its own unfolding.

In ED terms: Recursion is the ED regime where procedural structures become self-applicable, self-modifying, and self-directing.

This is the architectural meaning of recursion.

Recursion is the hinge between computational systems and autoprocedural architectures. It is the domain where:
- procedural form becomes self-applicable
- self-applicability becomes self-modification
- self-modification becomes architectural evolution
- architectural evolution becomes autoprocedural becoming

These capacities do not yet constitute reflection, meta-procedural reasoning, or self-interpreting architectures. But they form the organizational foundation from which all three will arise.

Paper 21 will develop the next threshold: how autoprocedural systems become reflective, how reflection becomes meta-procedural, and how meta-procedural architectures become the ED regime where becoming becomes self-interpreting.

8. Conclusion — Recursion as ED's First Self-Modifying Threshold

Recursion marks the moment where the architecture of becoming becomes self-directing. Computation gave the universe procedural structures — systems capable of executing symbolic transformations through explicit, repeatable, and deterministic operations. Recursion gives it self-applicable structures — systems capable of applying their own procedures to themselves, generating nested layers of execution, and producing unbounded procedural depth.

But recursion alone is not yet autonomy.

Self-application becomes autonomy only when recursive systems can modify their own procedures, stabilize those modifications, and govern the conditions under which modification occurs. Self-modification is the threshold where recursion becomes architecturally active, where recursive processes can transform the very structures that govern their own unfolding.

The recursive arc has shown that recursion arises when:
- computation becomes self-applicable
- self-application becomes self-reference
- self-reference becomes self-inspection
- self-inspection becomes self-continuation
- self-continuation becomes self-alteration
- self-alteration becomes persistent

- persistence becomes architectural
- architectural change becomes evolvable
- evolvability becomes autoprocedural

These transitions are not optional.
They are the structural consequences of systems that must not only execute procedures, but apply, interpret, extend, modify, and govern their own procedural architectures.

Recursion is the first ED regime where coherence is not merely conceptual, symbolic, or procedural, but self-procedural — where the architecture of becoming is shaped by structures that can apply themselves, transform themselves, and direct their own evolution.

In ED terms: Recursion is the ED regime through which procedural structures become self-applicable, self-modifying, and self-directing.

This is the architectural meaning of recursion.

Recursion is the hinge between computational systems and autoprocedural architectures. It is the domain where:

- procedural form becomes self-applicable
- self-applicability becomes self-modification
- self-modification becomes architectural evolution
- architectural evolution becomes autoprocedural becoming

These capacities do not yet constitute reflection, meta-procedural reasoning, or self-interpreting architectures. But they form the organizational foundation from which all three will arise.

Paper 21 will develop the next threshold: how autoprocedural systems become reflective, how reflection becomes meta-procedural, and how meta-procedural architectures become the ED regime where becoming becomes self-interpreting.