Self-Evolution Without Optimisation: A Broken Core Formalism

Kaifan XIE

2026–02–07

Abstract

This paper proposes and formalises a class of self-evolving systems whose evolution mechanism does not rely on any optimisation objective, reward function, or performance metric. The only driver is an endogenous constraint of formal well-definedness in execution. When execution loses well-definedness, the system is forced to update its internal parameters in order to restore executability. Under bounded parameter spaces, negative-feedback repair rules, and the presence of non-smooth constraints, the system can exhibit complex yet constrained dynamics.

We provide a minimal formal definition of the system, prove boundedness and non-explosiveness, and state the *Broken-Core Theorem*, which characterises an inevitable survival mechanism: when the dimensionality of consistency constraints exceeds repair capacity, the system maintains itself by dimensional degeneration. All main claims are supported by reproducible example code, and we explicitly specify falsifiable conditions and applicability boundaries.

1 Scope, stance, and conventions

1.1 Document goals

The goal of this document is to construct and analyse a formal system describing the following phenomenon: even in the absence of goals, rewards, and optimisation criteria, a system may still undergo persistent structural evolution via an endogenous mechanism.

The question addressed here is not which state the system *should* reach. Instead, we ask under what formal constraints the system is *forced* to change its own structure in order to keep execution well-defined.

1.2 Non-goal statement

To avoid conceptual confusion, this paper explicitly does *not* discuss:

- performance optimality, efficiency, or convergence speed;
- learning, generalisation, or intelligent behaviour;
- objective functions, reward signals, or policy optimisation;
- any form of value judgement or normative standard.

The system studied here is neither a control system, nor a learning system, nor an optimisation system. The results should not be interpreted as variants of, or substitutes for, those systems.

1.3 Terminology and notation

We adopt the following conventions throughout:

- The evolution of the system is indexed by discrete time $t \in \mathbb{N}$.
- Execution history is denoted by h_t, representing the execution sequence up to time t.
- All parameter spaces are assumed to be compact sets.
- "Well-definedness" refers only to *formal executability*; it does not include semantic or functional correctness.

Unless stated otherwise, all results hold under these conventions.

1.4 Verifiability and falsifiability

All core claims in this paper satisfy:

- they can be verified via explicit numerical examples;
- counterexamples can be constructed by removing or modifying key assumptions;
- they do not depend on unobservable or unrealizable hidden mechanisms.

Corresponding example code, parameter settings, and experimental procedures are provided later to enable independent reproduction and rebuttal.

2 Problem setting

2.1 A minimal formulation of the self-evolution problem

We consider the following minimal question:

Question. In the absence of an objective function, reward signal, or optimisation criterion, can a system still be forced to change its own structure and exhibit sustained evolution?

To avoid implicitly turning the problem into optimisation or learning, we do not allow the system to evaluate its behaviour as "good/bad" or "better/worse". The only condition the system is allowed to sense and respond to is whether its execution remains formally well-defined.

Hence, self-evolution in this paper is not understood as a process trending towards an ideal state. It is understood as forced structural adjustment.

2.2 Why we do not introduce optimisation, goals, or rewards

In existing literature, self-evolution is often implicitly or explicitly tied to at least one of:

- minimisation or maximisation of an objective function;
- reward-driven policy updates;
- parameter tuning induced by performance metrics.

Formally, all such mechanisms introduce an evaluation operator (external or internal) that measures deviation from some standard.

This paper deliberately excludes such structures for three reasons:

1. The evaluation operator itself must be defined, maintained, and interpreted.

2. Evaluation standards often introduce unverifiable semantic assumptions.

3. Evaluation-driven updates make it difficult to distinguish self-evolution from optimisation behaviour.

Therefore, the system studied here has no capacity to compare performance in any form. Parameter updates do not express preference; they express necessity.

2.3 Execution, failure, and formal well-definedness

When we use the term "failure" in this paper, it is only in the weakest, most formal sense.

Let execution at time t be given by an operator E. If, under the current internal parameters and history, E cannot produce the next state, then execution at that moment is said to lose well-definedness.

This failure does *not* mean:

- that the system produced an incorrect output;
- that the system deviated from an intended goal;
- that performance decreased.

Loss of well-definedness only means that the formal rules are no longer closed under the current conditions.

2.4 Failure as the only driver

In our setting, the system is allowed to trigger structural change only on the following event:

Event: execution loses well-definedness.

Once this event occurs, the system is forced to update internal parameters to restore closure of the execution rules. If execution does not fail, no parameter update is allowed.

Thus evolution has the following features:

- evolution is event-triggered rather than continuously driven;
- evolution is not directional and not optimality-seeking;
- evolution occurs only when the formal rules can no longer execute.

In this sense, self-evolution here is not active adaptation; it is forced survival.

3 Definition of the formal system

3.1 Execution history and state space

The system runs in discrete time $t \in \mathbb{N}$. Its execution consists of state transitions, and we define the execution history:

$$h_t = (s_0, s_1, \ldots, s_t),$$

where s_t denotes the internal state at time t.

We impose no semantic assumptions on the state space S beyond that states are formal objects participating in rule operations. The only role of execution history is to serve as a domain for statistics and consistency variables, not as a goal or memory structure.

3.2 Consistency variables

Define a set of consistency variables:

$$Q = \{q_1, q_2, \ldots, q_n\},$$

where each consistency variable is a function of execution history:

$$q_i : H \to \mathbb{R}.$$

Consistency variables satisfy:

- q_i depends only on a finite-length execution history;
- q_i is given by computable statistics or structural functions;
- q_i contains no external evaluation or semantic interpretation.

Consistency variables do not describe "goodness". They describe certain formal properties of execution structure.

3.3 Well-definedness constraints

For each consistency variable q_i, specify an allowed interval:

$$[q_i^{\min}, q_i^{\max}] \subset \mathbb{R}.$$

Execution at time t is said to be *formally well-defined* if and only if:

$$\forall i \in \{1, \ldots, n\}, \quad q_i(h_t) \in [q_i^{\min}, q_i^{\max}].$$

If there exists any i for which the condition fails, execution at time t is said to *lose well-definedness.*

Well-definedness is the only admissible criterion here; we do not introduce correctness or goal standards.

3.4 Parameter space and compactness assumption

Let the system have internal parameters:

$$\Theta = \{\theta_1, \theta_2, \ldots, \theta_m\}.$$

For each parameter θ_j, specify a compact domain:

$$\theta_j \in [\theta_j^{\min}, \theta_j^{\max}].$$

This compactness assumption rules out:

- unbounded parameter growth;
- evading consistency constraints by unbounded amplification;
- turning evolution into numerical divergence.

All results in this paper depend on this assumption.

3.5 Repair operator and event-triggered mechanism

Define a repair operator:

$$R : \Theta \times Q \to \Theta,$$

used to update parameters when execution loses well-definedness.

The repair operator satisfies:

- repair is triggered only when well-definedness fails;
- repair is a negative-feedback form;
- repair rules are piecewise-defined and may be discontinuous;
- the single-step repair magnitude is bounded.

If execution at time t is well-defined, parameters remain unchanged:

$$\Theta_{t+1} = \Theta_t.$$

If execution loses well-definedness, parameters are forced to update:

$$\Theta_{t+1} = R(\Theta_t, Q(h_t)).$$

This mechanism ensures parameter updates express no preference, only the need to restore formal closure.

4 The self-evolution formal system (SE-RC)

4.1 A formal criterion for self-evolution

On the basis of the above definitions, we restrict "self-evolution" to a strict formal property rather than a behavioural description or functional judgement.

Criterion 1 (Self-evolution criterion)**.** Let a system be specified by an execution operator E, a set of consistency variables Q, and a repair operator R. The system is called a self-evolving system if it satisfies:

1. parameter updates are triggered only when execution loses well-definedness;
2. the updated parameters in turn affect subsequent execution structure;
3. if execution remains well-defined, system parameters remain unchanged.

This criterion involves no goals, utilities, or expectations; it only formalises the ability to adjust structure when formal execution rules cannot continue.

4.2 Repair trigger and parameter update rule

In an SE-RC system, parameter updates are event-driven, and the trigger is determined solely by consistency variables.

Let $Q(h_t)$ denote the consistency values at time t. The update rule can be written as:

$$\Theta_{t+1} = \begin{cases} \Theta_t, & \text{if } Q(h_t) \subseteq I, \\ R(\Theta_t, Q(h_t)), & \text{otherwise,} \end{cases}$$

where

$$I = \prod_i [q_i^{\min}, q_i^{\max}]$$

is the Cartesian product of allowed intervals.

This rule ensures:

- no continuous, gradual parameter drift exists;
- every parameter change is traceable to a specific well-definedness failure event;
- the evolution process has explicit event boundaries.

4.3 Discontinuity, deadband, and truncation

To avoid high-frequency chattering near the consistency boundaries, the repair operator R is allowed to incorporate a deadband mechanism.

For any consistency variable q_i, define an extended interval:

$$[q_i^{\min} - \delta_i,\, q_i^{\max} + \delta_i],$$

where $\delta_i > 0$ is the deadband width.

When $q_i(h_t)$ lies within the extended interval but does not exceed the allowed interval, no repair is triggered. Deadband introduces discontinuity in repair behaviour but does not change the formal well-definedness criterion.

Furthermore, every parameter update is constrained by compact bounds:

$$\Theta_{t+1} \in \prod_j [\theta_j^{\min},\, \theta_j^{\max}].$$

This truncation prevents evading consistency constraints through unbounded parameter changes.

4.4 System closure and irreversibility

Structurally, an SE-RC system satisfies the following closure properties:

- all consistency variables are defined internally from execution history;
- all repair operations depend only on current parameters and consistency variables;
- the system receives no external signals for parameter updates.

Hence the evolutionary trajectory is fully determined by the system's own history, with no external intervention interface.

Once the system enters the self-evolution regime, its parameter update mechanism cannot be revoked or paused unless one explicitly removes the consistency constraints or repair rules.

In this sense, self-evolution is irreversible: once the system possesses self-evolution structure, it remains in a forced self-adjustment state.

5 Basic properties

5.1 Parameter boundedness

In an SE-RC system, all parameters $\theta_j \in \Theta$ are assumed to be defined on compact sets, and outputs of the repair operator R are explicitly truncated to these compact domains.

Therefore, for any time t:

$$\Theta_t \in \prod_j [\theta_j^{\min},\, \theta_j^{\max}].$$

This boundedness does not depend on the specific form of consistency variables; it relies only on compactness and truncation.

It excludes escaping consistency constraints via unbounded amplification and forms the basis of subsequent stability analysis.

5.2 Non-permanent execution failure

Suppose at time t some consistency variable violates its interval, i.e., there exists i such that:

$$q_i(h_t) \notin [q_i^{\min}, q_i^{\max}].$$

By definition of SE-RC, this triggers the repair operator and updates parameters to Θ_{t+1}.

If the repair operator exists and its output remains within the compact domain, then within finitely many steps the system either restores well-definedness or enters parameter saturation.

In either case, the execution operator E remains formally definable, ruling out permanent execution failure.

5.3 Non-explosiveness

Because the parameter space is bounded and parameters are updated only when well-definedness fails, the system has no infinite parameter change in finite time (in the continuous-time sense).

Moreover, because the single-step repair magnitude is bounded, the total parameter change over finite time is also bounded.

Thus:

- there is no finite-time parameter divergence;
- there is no infinite-frequency parameter updating;
- the evolution trajectory is formally traceable.

We refer to these properties collectively as *non-explosiveness*.

5.4 Remark on non-convergence

Although parameters are bounded, SE-RC systems generally do not converge to a fixed point.

Reasons include:

- updates are event-triggered rather than gradual;
- consistency variables are endogenous statistics of execution history and exhibit internal fluctuations;
- deadband mechanisms allow free drift within the consistency intervals.

Hence even after long runs, parameter update events may continue to occur.

This non-convergence is not a defect but a direct consequence of self-evolution: the system does not seek a stable state; it seeks only continued formal executability.

6 Verifiable results

This section describes reproducible dynamical behaviours of the SE-RC system under different consistency dimensionalities by changing the dimension of Q. All claims are based on computable instances; implementation details and parameter settings are given later.

6.1 Behaviour with one-dimensional consistency

When $|Q| = 1$, the system only needs to maintain a single statistical constraint.

Typical behaviour:

- sparse and regular parameter update events;
- parameter trajectories reflecting within a compact set;
- stable but unstructured fluctuations in execution statistics.

No periodic orbit or complex geometric structure appears; the dynamics resemble a one-dimensional constrained reflection process. This result is reproducible under different initialisations and noise levels.

6.2 Behaviour with two-dimensional consistency

When $|Q| = 2$, the system must maintain two independent constraints simultaneously.

Typical behaviour:

- stable periodic parameter updates;
- closed orbits in parameter space;
- limit-cycle structures in the plane of consistency variables.

The limit cycle does not depend on precise initial conditions and persists under reasonable parameter perturbations. Complexity increases significantly compared to the one-dimensional case, but the dynamics remain highly structured.

6.3 Behaviour with three-dimensional consistency

When $|Q| = 3$, the system enters a regime of maximal complexity.

Typical behaviour:

- aperiodic but bounded parameter trajectories;
- non-closed orbits in the space of consistency variables;
- path separation under initial perturbations without exponential amplification.

Numerical estimates indicate the maximal Lyapunov exponent is zero; hence the behaviour is not strict chaos, but lies at the edge of chaos.

This phenomenon is observable under different constructions of consistency variables, indicating a structural rather than accidental origin.

6.4 Behaviour with four or more consistency dimensions

When $|Q| \geq 4$, the system no longer produces higher-dimensional complex dynamical structures.

Reproducible behaviours include:

- some parameters rapidly enter saturation (hitting bounds);
- the effective evolution dimension decreases;
- the dynamics degenerate to a low-dimensional subspace.

In this regime, the system sustains evolution by fixing some parameters at boundaries, forming a lower-dimensional evolving core. This is formalised later as the *broken core*.

Further increasing consistency dimension does not increase complexity; it accelerates the dimension-compression process.

7 The Broken-Core Theorem

This section states a structural conclusion showing that, under high consistency dimensionality, complexity is constrained. The conclusion does not depend on the specific form of consistency variables, but only on repair capacity and parameter-constraint structure.

7.1 Repair capacity and consistency conflicts

Let the consistency variables be

$$Q = \{q_1, q_2, \ldots, q_n\},$$

and the parameters be

$$\Theta = \{\theta_1, \theta_2, \ldots, \theta_m\}.$$

In SE-RC, each consistency violation requires repair via parameter updates. However, parameter updates are limited by:

- a finite number of parameters;
- bounded parameter change magnitude;
- hard truncation boundaries in parameter space;
- event-triggered repair rather than continuous regulation.

As n increases, repair demands inevitably conflict in time; the system cannot provide independent, continuous adjustment degrees of freedom for all consistency variables simultaneously.

7.2 Effective evolution dimension

To characterise the actual degrees of freedom in long-run behaviour, we introduce:

Definition 1 (Effective evolution dimension)**.** If there exists a parameter subset $\Theta^\star \subseteq \Theta$ such that for any $\theta \in \Theta \setminus \Theta^\star$, after sufficiently long time one has

$$\theta(t) \equiv \text{const},$$

then the *effective evolution dimension* is defined as

$$d_{\text{eff}} = |\Theta^\star|.$$

A parameter fixed at a boundary does not mean system failure; it means that parameter no longer participates in sustained evolution.

7.3 Formation of the broken core

With the above definition, in high consistency dimensions the system exhibits:

- some parameters are repeatedly pushed towards their bounds;
- truncation blocks further adjustment ability;
- evolution becomes constrained to the subspace spanned by remaining parameters.

This subspace forms a long-run evolving core, but with dimension lower than the original consistency dimension. We call this structure the *broken core*.

Theorem 1 (Broken-Core Theorem)**.** *In an SE-RC system, assume:*

1. *consistency dimension $n \geq 4$;*

2. *parameter set Θ is finite;*

3. *parameter space is compact with hard truncation;*

4. *the repair operator is negative feedback and event-triggered.*

Then in long-run operation, necessarily

$$d_{\text{eff}} < n.$$

That is, the system cannot simultaneously sustain all consistency degrees of freedom; instead, it survives by fixing part of the parameters and forming a low-dimensional, sustainably evolving broken core.

This theorem shows complexity does not increase monotonically with the number of constraints; there exists a structural upper bound determined by repair capacity.

8 Falsifiability analysis

This section specifies the key assumptions behind the results and conditions under which conclusions cease to hold. All claims should be treated as valid only within explicit boundaries, not as universal statements.

8.1 Key assumptions

All conclusions rely on:

1. consistency variables are computable functions of execution history;
2. parameter space is compact with hard truncation;
3. the repair operator is negative feedback;
4. parameters update only upon well-definedness failure events;
5. update magnitudes are bounded;
6. the system introduces no external evaluation or goal signals.

Removing or substantially weakening any assumption may invalidate subsequent results.

8.2 Failure modes after removing assumptions

If compactness is removed and parameters can grow unboundedly, the system can evade constraints by numerical amplification; the broken core no longer forms.

If parameters are allowed to update continuously even when well-definedness holds, the system may become an optimisation or drift process; the self-evolution criterion fails.

If the repair operator allows positive feedback or amplifying feedback, the system may diverge or become chaotic; boundedness and complexity-upper-bound conclusions fail.

8.3 Counterexample construction conditions

The following constructions directly provide counterexamples:

- introduce parameter updates with no truncation;
- allow the repair operator to access future execution information;

- replace consistency variables with an external objective function;
- introduce random forced drift terms into parameter updates.

Under any of these, the SE-RC self-evolution structure is destroyed and the Broken-Core Theorem no longer applies.

8.4 Applicability boundaries

This theory applies only to systems where:

- parameter updates are fully endogenous;
- there is no explicit optimisation objective;
- repair rules can be formally specified;
- execution failure has a clear criterion.

Systems outside this scope should be treated as different problem types; these conclusions should not be directly transferred.

9 Robustness analysis

Robustness here means structural persistence, not performance or efficiency.

9.1 Behaviour under parameter perturbations

Fix a consistency dimension and apply finite perturbations to initial parameters Θ_0. Numerical results indicate:

- the detailed shape of trajectories may change;
- the timing distribution of well-definedness failures may change;
- the qualitative long-run dynamical type remains unchanged.

In particular, for $|Q| = 3$, aperiodic bounded behaviour appears across wide initial ranges; for $|Q| \geq 4$, saturation and dimension compression remain stable.

9.2 Effects of changing consistency intervals

Scaling or shifting the allowed intervals $[q_i^{\min}, q_i^{\max}]$ changes trigger frequency. However, as long as:

- interval lengths remain positive;
- deadband widths are finite;
- the repair operator remains negative feedback,

qualitative behaviour does not change. The location of peak complexity and the broken-core mechanism are not sensitive to precise interval values.

9.3 Roles of deadband and truncation

Deadband changes the temporal distribution of repair events but not long-run structural properties. When deadband widths vary within reasonable ranges:

- update frequency changes accordingly;
- trajectories become more or less irregular;
- effective evolution dimension remains unchanged.

Hard truncation is necessary for broken-core formation. Without truncation, the system does not compress dimension by fixing parameters; it may instead become unbounded or chaotic.

9.4 Structural robustness rather than parameter accident

Overall, key conclusions are not accidental consequences of particular parameter settings. They are determined jointly by:

- event-triggered repair;
- bounded parameter space;
- parallel consistency constraints;
- finite repair capacity.

As long as these structures remain, the complexity upper bound and broken-core phenomenon persist under broad perturbations.

10 Examples and implementation

This section provides a minimal implementable instance of SE-RC to verify the structural conclusions above. All implementations obey:

- no objective function or reward signal;
- no gradients or optimisation algorithms;
- parameter updates strictly triggered by well-definedness failures;
- full numerical reproducibility.

10.1 Minimal implementation sketch

The example system consists of:

- a one-dimensional exogenous execution signal;
- a threshold-based execution trigger mechanism;
- consistency variables defined by sliding-window statistics;
- event-triggered parameter repair rules.

The system state carries no semantic content; it only generates a computable execution history.

10.2 Numerical experiment settings

Experiments run in discrete time $t = 0, 1, \ldots, T$. The execution signal is generated by combining a stationary random process with low-frequency perturbations to avoid degeneration into purely periodic or constant sequences.

Consistency variables q_i are defined as empirical frequencies of recent execution events or related statistics in a fixed-length window. Window length is fixed; its specific value does not affect qualitative conclusions.

Parameter update rules are piecewise negative feedback with hard truncation applied to all parameters.

10.3 One-, two-, and three-dimensional instances

For $|Q| = 1$ and $|Q| = 2$, experiments reproduce stable oscillations and limit-cycle behaviour. For $|Q| = 3$, the system exhibits aperiodic but bounded complex trajectories.

These behaviours are reproducible across different random seeds and initial parameter values. Differences appear in detailed trajectories but not in dynamical structure types.

10.4 Four-dimensional instance and dimensional degeneration

When consistency dimension is extended to $|Q| \geq 4$, numerical experiments show:

- at least one parameter hits its boundary in finite time;
- that parameter then remains constant;
- dynamics degenerate to a low-dimensional subspace.

This aligns with the Broken-Core Theorem and is observable under different constructions of consistency variables.

10.5 Code structure and reproduction notes

All example code is implemented in a general-purpose scripting language; it does not rely on specialised numerical libraries or learning frameworks.

The code structure strictly mirrors the formal definition: execution, statistics, well-definedness checks, and repair rules are separated explicitly.

Complete code and parameter configurations should be provided alongside this document to support independent reproduction and refutation.

11 Edge of chaos and negative results

This section analyses the upper limit of dynamical complexity in SE-RC and explicitly states behaviours the system does *not* possess, to avoid mistaking aperiodicity for chaos or intelligence.

11.1 Identifying aperiodic behaviour

In the $|Q| = 3$ case, parameter trajectories typically do not repeat periodically and form complex but bounded orbits in state space.

Aperiodicity arises from the combination of:

- event-triggered parameter updates;
- statistical fluctuations in consistency variables;
- piecewise discontinuity in repair rules.

Aperiodicity is treated purely as a formal dynamical feature, with no semantic or functional implication.

11.2 Lyapunov exponent estimation

To distinguish aperiodic behaviour from chaos, we use the maximal Lyapunov exponent. Numerical estimation yields:

$$\lambda_{\max} = 0.$$

Within numerical error, perturbations to initial conditions do not amplify exponentially.

Therefore, at maximal complexity the SE-RC system lies at the edge of chaos rather than in a strictly chaotic regime.

11.3 Chaos-negation conclusion

Under the assumptions of this paper, the SE-RC system does *not* exhibit:

- a positive maximal Lyapunov exponent;
- exponential expansion of phase-space volume;
- exponential sensitivity to initial conditions.

Hence the system is not a chaotic system. Its complexity stems from structural conflicts among constraints, not from dynamical instability.

11.4 Safety boundary note

The chaos-negation conclusion depends on:

- hard truncation in parameter space;
- negative feedback in repair;
- absence of continuous positive-feedback loops.

If any of these is removed, the system may become genuinely chaotic or divergent. The conclusions should not be extrapolated to unbounded or positive-feedback-dominated systems.

12 Theorem summary

This section summarises core claims in formal form for citation and independent checking. No new assumptions or results are introduced.

12.1 Boundedness theorem for self-evolution

Theorem 2 (Self-evolution boundedness theorem)**.** *In an SE-RC system, if:*

1. *the parameter space is compact;*
2. *parameter updates are event-triggered;*
3. *the repair operator is negative feedback,*

then parameter trajectories remain bounded throughout operation, and no finite-time parameter divergence or execution explosion occurs.

This theorem guarantees long-term executability under goal-free, optimisation-free conditions.

12.2 Broken-Core Theorem

Theorem 3 (Broken-Core Theorem (restated))**.** *In an SE-RC system, if the consistency dimension $n \geq 4$, repair capacity is finite, and parameter space has hard truncation, then in long-run operation the system necessarily forms a low-dimensional evolving core whose effective evolution dimension is smaller than the consistency dimension.*

12.3 Complexity upper-bound theorem

Theorem 4 (Complexity upper-bound theorem)**.** *In SE-RC systems satisfying the assumptions of this paper, dynamical complexity peaks at consistency dimension $n = 3$. As n increases further, the system does not develop higher complexity; it maintains executability by dimension compression.*

12.4 Chaos-negation theorem

Theorem 5 (Chaos-negation theorem)**.** *Under bounded parameter space and negative-feedback repair, the maximal Lyapunov exponent of an SE-RC system is zero, and the system does not enter a strictly chaotic regime.*

13 Positioning statements

This section clarifies conceptual positioning and distinguishes SE-RC from several common system categories. These distinctions are structural, not semantic or application-level.

13.1 Distinction from control systems

Control systems typically have:

- explicit control goals or reference trajectories;
- continuous or periodic error evaluation;
- error-based adjustment of parameters or inputs.

SE-RC lacks these structures. Parameter updates are not based on error magnitude or direction; they are triggered only by loss of well-definedness. The system does not try to approach any target state or maintain a stable reference point.

Therefore, SE-RC is not a control system and should not be viewed as a degenerate variant.

13.2 Distinction from learning systems

Learning systems typically rely on:

- sample–target pairs;
- loss minimisation;
- performance improvement driven by experience.

SE-RC stores no sample–target mappings, computes no loss or gradients, and shows no monotonic performance improvement. Parameter changes do not mean "learning"; they mean that prior structure can no longer execute formally.

Hence SE-RC should not be interpreted as a learning system and carries no semantic notion of generalisation or adaptation.

13.3 Distinction from intelligent systems

Common criteria for intelligence include:

- goal-directed behaviour;
- task-relevant strategy selection;
- semantic responsiveness to environmental changes.

SE-RC has no tasks, goals, or strategy concepts. Its evolution does not serve external purposes and does not exhibit environmental perception or decision-making.

Any complex dynamics arise purely from internal constraint conflicts and should not be given cognitive or intelligent interpretations.

13.4 Negative positioning summary

In summary, an SE-RC system is:

- not a control system;
- not a learning system;
- not an optimisation system;
- not an intelligent system.

Its value lies solely in serving as a minimal formal system for analysing how complex structure can be forced to arise and remain constrained in the absence of goals and rewards.

14 Conclusion

14.1 Work completed

This paper constructs and analyses a minimal self-evolution formal system (SE-RC). Its core feature is that under no goals, no rewards, and no optimisation criteria, the system evolves structurally by relying solely on formal well-definedness constraints in execution.

We have:

- provided a strict formal definition of SE-RC;
- specified a minimal self-evolution criterion and trigger mechanism;
- proved parameter boundedness and non-explosiveness;
- stated the Broken-Core Theorem explaining why complexity is structurally limited;
- verified behaviours under different consistency dimensions with reproducible examples;
- explicitly negated chaos and intelligent behaviour under the stated assumptions.