Research Article *Open Access*

Journal of Pioneering Artificial Intelligence Research

ISSN: 3069-0846

DOI: *doi.org/10.63721/26JPAIR0130*

Deterministic Autonomous Structural Elimination Framework

Huseyin Murat Cekirge

Department of Mechanical Engineering, The City College of New York (CUNY), New York, USA

Citation: *Huseyin Murat Cekirge (2026) Deterministic Autonomous Structural Elimination Framework. J of Poin Artf Research 2(2), 1-15 WMJ-JPAIR-130*

Abstract

This study introduces a deterministic framework for autonomous decision-making based on structural elimination over finite candidate sets. In contrast to conventional approaches in robotics and machine learning, where decisions are obtained through iterative search, optimization, or learning, the proposed framework formulates autonomy as a non-iterative process in which infeasible candidates are progressively removed under deterministic objectives. The system operates within an open objective architecture, where each objective acts as a constraint and is admitted only if feasibility is preserved, ensuring structural consistency while allowing extensibility of goals. The framework is further situated within a three-layer view of autonomous systems, comprising perception, decision, and execution. At the perception level, recognition is achieved through Structural Elimination Recognition (SER), while at the decision level, admissible actions are obtained via constraint-based pruning of candidate sets. Execution is treated as a separate layer governed by system dynamics and control, where iterative methods may still be required. A minimal illustrative example demonstrates that decisions can be revealed without search or parameter updates, even under changing objective conditions in real practical scenarios. The results highlight a complementary regime of autonomy in which solutions emerge from the elimination of infeasible alternatives rather than exploration of a solution space. The framework reveals that the limiting factor in autonomous systems is the compatibility structure of the objective set.

***Corresponding author**: Huseyin Murat Cekirge, Department of Mechanical Engineering, The City College of New York (CUNY), New York, USA. E-mail: hmcekirge@usa.net

Submitted: 20.03.2026 **Accepted**: 24.03.2026 **Published**: 05.04.2026

Keywords: Deterministic Decision-Making, Constraint-Based Systems, Autonomous Systems, Structural Elimination, Non-Iterative Methods, Discrete Decision Processes, Objective Compatibility, Open Objective Architecture

Introduction

Autonomous systems have become a central paradigm in modern computational and engineering practice, spanning applications from robotics and control to large-scale data-driven decision systems. In prevailing formulations, autonomy is closely associated with the ability of a system to learn from data, adapt through experience, and optimize

performance over time. These capabilities are typically realized through iterative procedures such as gradient-based optimization, reinforcement learning, or probabilistic inference, where system behavior emerges from successive parameter updates guided by objective or reward functions [1-3].

Within this dominant framework, decision-making is inherently process-driven. A system transitions through a sequence of intermediate states, gradually approaching an optimal or satisfactory solution. The quality of the final outcome depends on factors such as initialization, convergence properties, data availability, and hyperparameter selection. While these approaches have demonstrated remarkable success in complex and uncertain environments, they introduce dependencies on training procedures, computational resources, and stability considerations [4-5].

In parallel to these developments, earlier and alternative perspectives have emphasized decision and recognition problems, where solutions are derived through direct evaluation of constraints or compatibility conditions rather than through iterative improvement [6]. Such approaches, although less prominent in contemporary large-scale systems, highlight an important conceptual distinction: decision outcomes can, in principle, be obtained through finite elimination of infeasible candidates, without recourse to learning or optimization trajectories.

Motivated by this distinction, the present study investigates a minimal autonomous framework based on a deterministic and extensible objective structure. The goal is not to replace learning-based systems, but to isolate and formalize a complementary mode of autonomy in which:

- decision-making is non-iterative,
- objectives act as deterministic constraints,
- system behavior is governed by the structure of admissible solutions.

A key question arises in such a setting: if the objective set is allowed to grow, what governs the stability and operability of the system? In conventional formulations, additional objectives are typically handled through weighting, aggregation, or multi-objective optimization. In contrast, this work proposes that objective integration can be governed by a simple structural principle: preservation of feasibility.

The contribution of this paper is therefore twofold. First, it introduces a toy autonomous model with an open objective architecture, designed to capture the essential mechanisms of autonomous operation without reliance on stochastic or iterative processes. Second, it establishes an admissibility rule for objective extension, showing that the limiting factor in such systems is the compatibility structure of the objective set.

In this work, a Deterministic Autonomous Structural Elimination Framework is introduced, in which decision-making is achieved through the progressive elimination of infeasible candidates under deterministic objectives. The proposed framework can be interpreted as a decision-level extension of the Structural Elimination Recognition (SER) paradigm, in which the same structural elimination principle is applied to candidate actions under deterministic objectives.

Together, SER and the present framework suggest a unified Structural Elimination Paradigm in which both perception and decision-making are governed by deterministic reduction of finite candidate sets rather than iterative optimization.

Under this view, both recognition and decision-making are governed by a common structural mechanism: a finite candidate set is progressively reduced through constraint evaluation until a consistent solution remains. No iterative optimization, parameter learning, or stochastic search is required; instead, solutions are revealed through compatibility structure. This establishes a unified perspective in which perception and action are not fundamentally different processes, but instances of a single structural principle operating over different domains.

This perspective suggests a broader Structural Elimination Paradigm, in which intelligent behavior emerges from deterministic reduction processes applied to well-defined candidate sets. Within this paradigm, recognition corresponds to identifying a consistent representation, while decision-making corresponds to selecting or constructing a feasible

action. The distinction lies not in the underlying mechanism, but in the nature of the candidate space and the constraints imposed upon it. This work demonstrates that autonomy can arise without optimization or learning, instead emerging from the structured elimination of infeasible candidates.

Model Definition

From a computational standpoint, the proposed framework replaces iterative search in parameter space with direct evaluation over a finite candidate set. Each objective induces a deterministic filtering operation, reducing the feasible set without requiring convergence procedures or repeated updates. As a result, computational effort scales with the number of candidates and objectives, rather than with the dynamics of an optimization process.

System Structure

Consider a finite set of candidates:

$$X = \{x_1, x_2, \ldots, x_n\} \quad (1)$$

Each element $x \in X$ represents a possible decision, configuration, or action available to the system.

Let $O = \{O_1, O_2, \ldots, O_k\}$ denote a set of objectives. In the present framework, each objective O_i is defined as a deterministic constraint acting on X:

$$O_i : X \rightarrow \{0, 1\} \quad (2)$$

where O_i is interpreted as an indicator function of feasibility:

- $O_i(x) = 1$ indicates that candidate x satisfies the objective,
- $O_i(x) = 0$ indicates violation.

Feasible Set

The system state is represented by the feasible set:

$$F(O) = \{ x \in X \mid O_i(x) = 1 \text{ for all } i = 1, \ldots, k \} \quad (3)$$

This set contains all candidates that simultaneously satisfy all admitted objectives. Decision-making is therefore not formulated as an optimization trajectory, but as a set reduction process, where infeasible candidates are eliminated through successive constraint evaluation.

Objective Integration

The architecture is open in the sense that new objectives can be introduced dynamically. Let O_{k+1} be a new objective. Its integration induces an updated feasible set:

$$F_{new} = \{ x \in F(O) \mid O_{k+1}(x) = 1 \} \quad (4)$$

Thus, each additional objective refines the solution space through further elimination.

Admissibility Rule

To ensure that the system remains operational, objective extension is governed by the following rule:

A new objective is admitted only if the resulting feasible set is non-empty.

Formally,

$$O_{k+1} \text{ is admissible} \Leftrightarrow F_{new} \neq \emptyset \quad (5)$$

If $F_{new} = \emptyset$, the objective must be either:

- rejected, or
- relaxed, or
- deferred for later evaluation.

This condition defines the structural stability of the system under objective growth.

Decision Rule

A decision is obtained by selecting an element from the feasible set:

$$x^* \in F(O) \quad (6)$$

If $|F(O)| = 1$, the solution is uniquely determined.
If $|F(O)| > 1$, additional objectives may be introduced to refine the set.
If $|F(O)| = 0$, the system has reached an infeasible state, indicating incompatibility among objectives.

Interpretation

The proposed model can be interpreted as a **deterministic autonomous system** with the following characteristics:

- Autonomy arises from **automatic constraint evaluation**,
- Adaptation is replaced by **objective extension**,
- Computation is finite and **non-iterative in principle**,
- System behavior is governed by the **structure of admissible objectives.**

In contrast to optimization-based systems, where solutions are approached through iterative improvement, the present framework reveals solutions through successive elimination of incompatible candidates. Consequently, the primary limitation of the system is not computational complexity, but the compatibility of the objective set.

Numerical Toy Example

Problem Setup

Consider a finite candidate set:

$X = \{x_1, x_2, x_3, x_4, x_5\}$ (7)

Each candidate represents a possible decision option characterized by four attributes:

- Cost (C)
- Time (T)
- Comfort (K)
- Directness (D)

The values are given in Table 1.

Table 1: Summarizes the candidate set and associated attributes, including cost (C), time (T), comfort (K), and directness (D).

Candidate	Cost (C)	Time (T)	Comfort (K)	Direct (D)
x_1	500	5	3	1
x_2	300	8	2	0
x_3	450	6	4	1
x_4	350	7	3	0
x_5	600	4	5	1

Objective Definition

Each objective is formulated as a deterministic constraint:

- O_1: Cost ≤ 500
- O_2: Time ≤ 6
- O_3: Comfort ≥ 3
- O_4: Directness $= 1$

Stepwise Elimination Process

Step 1 — Apply O_1 (Cost ≤ 500)

Feasible set:

$F_1 = \{x_1, x_2, x_3, x_4\}$ (8)

(x_5 eliminated)

Step 2 — Apply O_2 (Time ≤ 6)

$F_2 = \{x_1, x_3\}$ (9)

(x_2 and x_4 eliminated)

Step 3 — Apply O_3 (Comfort ≥ 3)

$F_3 = \{x_1, x_3\}$ (10)

(no elimination)

Step 4 — Apply O_4 (Direct = 1)

$F_4 = \{x_1, x_3\}$ (11)

(no elimination)

Decision State

The final feasible set is:

$F = \{x_1, x_3\}$ (12)

Since multiple candidates remain, the system may:

- accept the solution set, or
- introduce an additional objective.

Objective Extension

Introduce a new objective:

- O_5: Time ≤ 5

Apply O_5:

$F_5 = \{x_1\}$ (13)

Result

A unique solution is obtained:

$x^* = x_1$ (14)

Inadmissible Objective Example

Consider an alternative objective:

- O_6: Cost ≤ 300

Apply O_6 to $F_4 = \{x_1, x_3\}$:

$F_{new} = \emptyset$ (15)

Interpretation

- O_6 eliminates all candidates
- The system reaches an infeasible state

According to the admissibility rule:

O_6 is rejected (or must be relaxed)

Observations

This example illustrates the core properties of the proposed framework:

1. **Finite Elimination**

The solution is obtained through successive filtering, without iterative search or optimization.

2. **Objective-Driven Refinement**

Each objective reduces the feasible set, increasing specificity.

3. Admissibility Constraint

Not all objectives can be accepted, feasibility governs system stability.

4. Open Architecture

Additional objectives can be introduced dynamically to refine decisions.

Computational Interpretation

The entire process consists of:

- finite evaluations of constraints
- set intersection operations

No convergence process, parameter update, or trajectory search is required.

This toy example demonstrates that autonomous decision-making can be realized as a finite structural elimination process, where solution existence and uniqueness are governed by objective compatibility rather than iterative optimization.

Comparison with Optimization-Based Approaches

In conventional optimization-based frameworks, decision-making is formulated as the search for an optimal solution through iterative improvement of an objective function. Methods such as gradient-based optimization or reinforcement learning generate a sequence of intermediate states, gradually approaching a solution based on convergence criteria, learning rates, and data-dependent updates. As a result, computational effort is largely determined by the dynamics of the search process and the properties of the objective landscape.

In contrast, the proposed deterministic framework does not construct a trajectory toward a solution. Instead, it evaluates a finite set of candidates under a collection of objectives, where each objective acts as a constraint that eliminates incompatible options. The solution, when it exists, is revealed through successive reduction of the feasible set rather than through asymptotic convergence.

This distinction leads to a fundamental difference in computational structure: optimization-based methods expend computation exploring a solution space, whereas the present framework expends computation discarding infeasible elements from a predefined set. Consequently, the two approaches are not competing formulations but complementary perspectives, applicable under different problem regimes. Optimization seeks a solution through motion in a search space; deterministic elimination reveals a solution through structure.

Telescopic Growth (Sequential Extension)

Objectives are added along a single decision chain, progressively refining the feasible set.

- $O_1 \rightarrow O_2 \rightarrow O_3 \rightarrow \ldots$
- Each step reduces F

Interpretation

- Increasing resolution
- Narrowing toward a unique solution

Independent Seeding (Parallel Extension)

New objective sets can be initiated independently, forming parallel decision branches over the same candidate space.

- Branch A: $\{O_1, O_2, O_3\}$
- Branch B: $\{O_1, O_4, O_6\}$
- Branch C: $\{O_2, O_5\}$

Each produces its own feasible set:

- F_A, F_B, F_C

Significance of Dual Growth Modes

Two modes of growth:

Table 2: Modes of objective growth.

Mode	Type	Effect
Telescopic	Sequential	Deep refinement
Seeding	Parallel	Exploratory diversity

Open Architecture and Telescopic Growth

The proposed framework admits an open objective architecture in which system evolution can occur through two complementary mechanisms. First, telescopic growth refers to the sequential extension of an objective set, where each newly admitted objective further refines the feasible solution set through deterministic elimination. Second, independent seeding allows the initiation of parallel objective subsets, each defining a distinct evaluation pathway over the same candidate space. These parallel branches operate independently, producing multiple feasible sets corresponding to different objective

configurations.

This dual structure enables both progressive refinement and exploratory diversification without requiring iterative search or stochastic processes. System behavior is thus governed by the structure and interaction of objective subsets rather than by trajectory-based optimization. In this sense, optimization-based methods explore a solution space through iterative trajectories, whereas the proposed framework prunes the space directly by eliminating infeasible candidates under deterministic constraints. Optimization explores a solution space; the proposed framework resolves it through deterministic elimination.

Conceptual Example: The Closed Room Problem

Motivation

To illustrate the operational principles of the proposed deterministic autonomous framework, consider a simple but structurally representative scenario. The purpose of this example is not to model physical behavior in detail, but to demonstrate structural decision formation.

The example highlights three essential aspects of autonomy:

- initiation of a goal,
- internal generation of candidate actions, and
- progressive elimination of infeasible options under structural constraints.

Problem Description

A person is located inside a room with a closed door and seeks to exit. The state of the door (locked or unlocked) is initially unknown.

The primary objective is defined as:

- O_1: exit the room

This objective arises naturally from the current state and does not require external specification beyond the recognition of a constraint (being confined within the room).

Candidate Generation

The system generates a finite set of possible actions:

$X = \{$
x_1: turn the door handle,
x_2: search for a key,
x_3: knock on the door,
x_4: attempt an alternative exit (e.g., window),
x_5: force the door open
$\}$

This set is not pre-enumerated by an external agent, but emerges from the internal structure of the system. This reflects a fundamental aspect of autonomy: the ability to propose candidate actions without prior optimization or training.

Objective Structure

Additional objectives are introduced to guide decision-making:

- O_2: action must be feasible under current conditions
- O_3: minimize effort
- O_4: avoid damage

Each objective acts as a deterministic constraint, partitioning the candidate set into admissible and inadmissible elements.

Sequential Elimination (Telescopic Growth)

The decision process proceeds through successive application of objectives.

Step 1 — Feasibility (O_2)

All candidate actions are initially considered feasible under uncertainty, so no elimination occurs.

Step 2 — Effort Minimization (O_3)

High-effort actions such as forcing the door or attempting structural escape are deprioritized or eliminated:

$$F_1 = \{x_1, x_2, x_3\} \quad (16)$$

Step 3 — Effectiveness (O_1)

At this stage, the evaluation depends on the actual state of the door:

- If the door is unlocked, turning the handle (x_1) satisfies the objective directly.
- If the door is locked, x_1 fails and is eliminated.

This produces two possible feasible sets:

- $F_{unlocked} = \{x_1\}$
- $F_{locked} = \{x_2, x_3\}$

Objective Extension and Information Resolution

In the presence of uncertainty, a new implicit objective emerges:

- O_5: reduce uncertainty about system state

The action x_1 (turning the handle) now serves a dual role:

- it may directly achieve the primary objective, and
- it provides information about whether the door is locked.

Thus, evaluation and information acquisition are integrated within the same structural process, without requiring a separate inference mechanism.

Parallel Branching (Independent Seeding)

Depending on the outcome of x_1, the system transitions into different branches:

- If successful → termination
- If unsuccessful → continue with $\{x_2, x_3\}$

Each branch corresponds to an independently evolving objective subset, consistent with the notion of independent seeding introduced earlier.

Interpretation

This example demonstrates several key properties of the proposed framework:

1. **Goal Initiation from State**
   The objective emerges directly from the system's condition, not from an externally imposed optimization function.
2. **Internal Candidate Generation**
   The system constructs its own action set, prior to evaluation.
3. **Deterministic Elimination**
   Decisions arise through filtering rather than through iterative improvement or search.
4. **Admissibility and Branching**
   Multiple feasible paths may exist, leading to parallel decision structures.
5. **Integration of Action and Information**
   Certain actions simultaneously reduce uncertainty and advance toward the objective.

Concluding Remark

The closed room example illustrates that autonomous decision-making can be formulated as a finite structural process in which actions, objectives, and information are unified within a deterministic elimination framework. The system does not search for an optimal solution; it reveals a valid one by progressively excluding incompatible alternatives.

Relational Extension: From Single Objective to Linked Actions

Multi-Stage Objective

Instead of a single objective:

- O_1: exit room

We now define a higher-level objective:

- O^*: reach home

Decomposition into Sub-Objectives

O^* can be decomposed into:

- O_1: exit room
- O_2: reach street
- O_3: obtain transportation (taxi)
- O_4: travel to home

These are not independent. They form a dependency chain.

Relational Structure

Define a relation:

$$R \subseteq X \times X \qquad (17)$$

where:

- $x_i \rightarrow x_j$ means action x_j is feasible only after x_i

Example:

- exit room → search taxi
- search taxi → take taxi
- take taxi → reach home

Candidate Structure Becomes a Graph

Previously:

- flat set: X

Now:
Structured set (graph or network)

- nodes = actions
- edges = feasible transitions

Feasibility Becomes Conditional

Before:

- x is feasible or not

Now:

- x is feasible given current state

Example:

- "take taxi" is NOT feasible unless "reach street" is satisfied

Deterministic Elimination Still Holds

Your framework still applies:
At each stage:

- evaluate feasible actions
- eliminate invalid ones

But now:
Applied locally per state

Sequential Feasible Path

Instead of:

- selecting one x*

You now obtain:
A sequence
$x_1 \rightarrow x_2 \rightarrow x_3 \rightarrow \ldots \rightarrow x_k$ (18)

Strong Insight

Autonomy extends from selecting a feasible element to constructing a feasible sequence under relational constraints.

No Optimization Still Needed

Important:

- No shortest path required
- No global optimization required

Only:

- local admissibility + relational consistency

The framework can be extended from single-stage decision problems to relational structures in which objectives are decomposed into dependent sub-objectives. In this setting, candidate actions are no longer independent elements but nodes in a directed structure defined by feasibility relations. Each action is admissible only within a specific state context, and decision-making proceeds through the construction of a feasible sequence rather than the selection of a single element. This preserves the deterministic elimination principle while enabling multi-stage autonomous behavior.

In the relational extension of the proposed framework, the candidate space is no longer represented as a single set but as a collection of interconnected feasible sets corresponding to different system states. These sets may partially overlap, with certain actions belonging to multiple contexts. The availability of an action is therefore not absolute but conditional on the current state and previously satisfied objectives. Decision-making proceeds by selecting admissible elements within the active set and transitioning to a new set, preserving the deterministic elimination principle at each stage. Actions do not exist globally, they exist within contexts defined by state and objective compatibility. The candidate action set may include not only direct physical actions but also relational actions involving external agents. For example, when a required resource is inaccessible, the system may generate an action that requests assistance from another agent. Such actions remain consistent with the deterministic framework, as they are evaluated through the same admissibility criteria. Autonomy is therefore not restricted to isolated operation, but may incorporate structured interactions with the environment and other agents.

The Situation Described

- Child wants to leave
- Cannot access key
- Says: “mama, give me the key”

This is not failure. This is an admissible action.

What This Means in Your Framework
Previously:

- actions = physical actions (turn handle, search, etc.)

Now you add:
relational actions involving other agents
Example:

- x_6: request key from mother
- Very Important Insight:

Asking for help is not outside autonomy — it is part of it.

Extend Candidate Set
Now:
X = {
turn handle,
search key,
knock,
force door,
request key (external agent)
}

New Type of Objective

You now implicitly introduce:

- O_6: minimize effort
- O_7: use available resources

If "mother exists and can help":

- requesting key becomes highly admissible

This is NOT Iteration — Still

The system does not:

- try repeatedly
- optimize

It does:
Selects a structurally valid action

Eyeglass Example (Very Relevant)

Same structure:

- cannot see
- instead of searching blindly

ask someone: "where are my glasses?"

Your system now includes:

Agent-Augmented Actions

Actions can be:

1. physical
2. informational
3. relational (involving others)

Autonomy does not imply isolation; it includes the ability to invoke external resources when structurally admissible.

Why This Is Powerful
You just solved a classical limitation:

- many systems assume closed world
- you naturally opened it

without breaking structure. Requesting assistance is not a breakdown of autonomy, but an admissible action within an extended decision structure.

How humans actually act:

And your system can describe it without learning, without iteration, without optimization. The proposed system shares a superficial structural resemblance to sequential models such as recurrent neural networks, in the sense that decisions evolve over successive states. However, it differs fundamentally in that no parameter learning or weighted aggregation is involved; all transitions are governed by a deterministic admissibility condition

Relational Objective Structures and Contextual Clustering

Motivation

In practical decision-making, objectives rarely appear in isolation. Human reasoning operates through interconnected concepts such as actions, locations, resources, and intentions. For example, the act of going is not a standalone objective, but is inherently linked to questions such as where to go, how to go, and *what is required to get there*. Similarly, activities such as eating are embedded within contextual structures involving time, location, and purpose.

This section extends the deterministic framework by introducing a relational organization of objectives, where decision-making is governed by interacting subsets rather than a single flat structure.

Input–Output Structure

The system can be viewed in terms of inputs, internal structure, and outputs:

- Inputs: -- Current state (e.g., location, availability of resources)
  -- Active objective (e.g., go home, eat lunch)
- Output: A feasible action or sequence of actions satisfying the objective

Formally, let:

- S denote the current state
- O denote the active objective
- X(S, O) denote the candidate action set

The system produces:

$$x^* \in F(S, O) \qquad (19)$$

where F(S, O) is the feasible subset determined by admissibility constraints.

Objective Clusters

Objectives naturally form clusters based on semantic and functional relationships.

Example 1: Movement Cluster ("GOING")

- Core objective: Omove = "go"
- Associated sub-objectives:

- destination: school, home, office
- origin: current location
- mode: walking, taxi, bus
- constraints: time, cost, accessibility

This defines a cluster:
C_move = {destination, origin, vehicle, constraints}

Example 2: Food Cluster ("EATING")
- Core objective: O_{food} = "eat"
- Associated sub-objectives:
- type: lunch, dinner, snack
- context: party, work, home
- location: restaurant, kitchen
- time: noon, evening

This defines:
C_{food} = {type, context, location, time}

Shared and Unshared Elements

Clusters are not independent. They may share elements.

Shared Example:
- "go to restaurant" involves both:
- movement cluster
- food cluster

Thus:
$C_{move} \cap C_{food} \neq \emptyset$ (20)

Unshared Example:
- "unlock door" belongs only to:
- access/control cluster

Relational Structure

The system can be interpreted as a collection of sets:
$\{C_1, C_2, \ldots, C_k\}$ (21)

with:
- partial overlaps
- conditional connections

Each cluster defines its own feasible set:
$F_i(S)$ (22)

and the active feasible set becomes:

$F(S) = \cap_i F_i(S)$ over active clusters

Conditional Activation

Not all clusters are active at all times.

Example:
- If the objective is "go home"
→ activate movement cluster
- If the person becomes hungry
→ activate food cluster

Clusters are therefore:
Contextually activated

Resource and Access Relations

Certain actions depend on resources:
- door → requires key
- taxi → requires availability
- food → requires access/location

These introduce binary conditions:
- key ∈ {0,1}
- door ∈ {open, closed}
- taxi ∈ {available, unavailable}

These conditions directly affect feasibility:
F(S) depends on resource states

Sequence Formation

When clusters interact, decisions form structured sequences:

Example:
- open door → exit → reach street → get taxi → go home

This is not derived through optimization, but through:
- cluster activation
- constraint satisfaction
- admissible transitions

Interpretation

The relational cluster structure introduces:
- Contextual reasoning

Decisions depend on active clusters and state
- Shared structure

Actions may belong to multiple clusters
- Conditional feasibility

Availability depends on resource state
- Structured sequences

Multi-stage behavior emerges naturally

Concluding Remark

Autonomous behavior can be interpreted as the navigation of a system of interconnected objective

clusters, where decisions arise from the interaction of shared and context-dependent constraints rather than from isolated optimization processes. Here, there is no claim to resolve the enigma of life; however, we may have clarified, at least in part, how certain decisions become possible and how, in simple terms, the door opens.

The framework also applies to agents with limited internal action capabilities. For example, a child attempting to leave a closed room may not possess the means to unlock or open the door directly. In such cases, the candidate set is restricted, and relational actions such as signaling or requesting assistance become dominant. The objective is still achieved through admissible actions, but the solution path incorporates external agents. This demonstrates that autonomy is relative to the available action space rather than to the complexity of the agent.

Computational Three-Layer Formulation (Perception–Decision–Execution)

The autonomous system can be formalized as a composition of three computational layers operating on structured sets: perception, decision, and execution. Each layer transforms a candidate set through deterministic or dynamic operators.

Perception Layer (SER)

Let

- Y denote the set of candidate representations,
- $C = \{C_1, C_2, \ldots, C_m\}$ denote structural compatibility constraints.

Define the feasible representation set:

$$R = \{ y \in Y \mid C_j(y) = 1 \text{ for all } j \} \quad (23)$$

Recognition is defined as:

$$y^* \in R \quad (24)$$

If $|R| = 1$, recognition is unique; otherwise, ambiguity remains.

Interpretation:

Perception is a deterministic filtering operator:

$$\Phi : Y \rightarrow R \quad (25)$$

Decision Layer (Deterministic Autonomous Structural Elimination Framework)

Let

- X denote the set of candidate actions,
- $O = \{O_1, O_2, \ldots, O_k\}$ denote objectives.

Define the feasible action set:

$$F = \{x \in X \mid O_i(x) = 1 \text{ for all } i \} \quad (26)$$

Decision is defined as:

$$x^* \in F \quad (27)$$

with admissibility condition:

$$F \neq \emptyset \quad (28)$$

Decision-making is therefore a deterministic reduction operator:

$$\Psi : X \rightarrow F \quad (29)$$

Execution Layer (Dynamics / Control)

Let

- S denote the system state space,
- $x^* \in F$ denote the selected action.

Execution is defined as a state transition:

$$s_{t+1} = T(s_t, x^*) \quad (30)$$

where T is a transition operator governed by system dynamics and control laws.

Unlike the previous layers, T may require iterative computation due to:

- continuous state space
- feedback control
- uncertainty

Layer Composition

The full system can be written as:

$$s_{t+1} = T(s_t, x^*) \quad (31)$$

or equivalently:

$$Y \rightarrow \Phi \rightarrow R \quad (32)$$

$$X \rightarrow \Psi \rightarrow F \quad (33)$$

$$(s_t, x^*) \rightarrow T \rightarrow s_{t+1} \quad (34)$$

Structural Distinction

The first two layers share a common computational form:

finite set + deterministic constraints → set reduction

The execution layer differs:

state + action → state evolution

Key Insight

Perception and decision are elimination operators over finite sets, while execution is a transition operator over a state space. This distinction can be illustrated through a simple example. A robot is placed in front of a closed door. Its objective is simple: to open the door.

Illustrative Example: Deterministic Door Opening

Consider a robot positioned in front of a closed door. The objective is to open the door and exit.

Define a finite candidate action set:
$X = \{$
x_1: turn handle,
x_2: push door,
x_3: pull door,
x_4: request key
$\}$

Define deterministic objectives:
O_1: door must open
O_2: action must be feasible
O_3: avoid damage

Each objective acts as a constraint:
$O_i : X \rightarrow \{0, 1\}$ (35)

Step 1 - Feasible Set
The feasible set is defined as:
$F = \{ x \in X \mid O_i(x) = 1 \text{ for all } i \}$ (36)

Step 2 - Evaluation
Case A: Door is unlocked
$F = \{ x_1 \}$ (37)
→ turning the handle satisfies all objectives
Case B: Door is locked
$F = \{ x_4 \}$ (38)
→ requesting the key is the only admissible action

Step 3 - Decision
$x^* \in F$. (39)

Interpretation

The solution is obtained by eliminating infeasible actions, not by searching or optimizing over trajectories. The robot does not explore a space; it evaluates a finite set and retains only admissible elements.

"The door is opened not by search, but by elimination."

The solution is what remains after the impossible is removed. When all infeasible actions are removed, the decision reveals itself. The system does not search for a solution; it first identifies the structure, then eliminates the impossible.

Comparison with Robotics-Based Approaches

Autonomous systems in robotics are typically formulated as problems of planning and control over continuous state and action spaces. In such settings, decision-making is achieved through search, optimization, or feedback-driven updates. Motion planning algorithms (e.g., sampling-based planners), optimal control methods, and learning-based policies operate by exploring a solution space and progressively improving candidate trajectories. As a result, computation is inherently iterative and often depends on convergence properties, model fidelity, and environmental uncertainty.

In contrast, the framework proposed in this study operates under a different structural assumption. Rather than exploring a continuous space, the system is defined over a finite set of candidate actions, and decision-making is achieved through the deterministic elimination of infeasible elements under a set of objectives. No trajectory generation, gradient-based update, or iterative refinement is required; instead, solutions are revealed through constraint compatibility within a discrete candidate structure.

This distinction reflects two complementary regimes of autonomous behavior. Robotics-based approaches are well-suited for low-level control and motion generation, where the state space is continuous and precise actuation is required. The proposed framework, by contrast, operates at the decision layer, where candidate actions can be explicitly enumerated and evaluated. In such settings, the primary challenge is not navigation through a space, but the identification of a structurally admissible option.

Importantly, the two approaches are not competing but complementary. A robotics system may employ iterative planning and control to execute an action, while the selection of that action can be governed by a deterministic elimination process. In this sense, the present framework can be viewed as providing a

structural decision layer that precedes and informs conventional robotic execution mechanisms.

Thus, autonomy may be decomposed into two interacting layers: a structural decision layer based on elimination, and an execution layer based on planning and control.

A Three-Layer Structural View of Autonomous Systems

The proposed framework can be situated within a broader structural view of autonomous systems consisting of three interacting layers: perception, decision, and execution. Each layer operates on a distinct domain, yet all can be interpreted through a common elimination-based principle.

Perception Layer (SER)

At the perception level, the Structural Elimination Recognition (SER) paradigm operates on a finite set of candidate representations. Recognition is achieved by progressively eliminating incompatible candidates based on structural compatibility constraints until a consistent representation remains.

In this layer:

- candidate set: possible object representations
- constraints: structural compatibility conditions
- outcome: recognized object

The process is deterministic and non-iterative, relying on elimination rather than optimization.

Decision Layer (Deterministic Autonomous Structural Elimination Framework)

At the decision level, the same structural principle is extended to action selection. The system operates over a finite set of candidate actions, and objectives act as deterministic constraints that eliminate infeasible options.

In this layer:

- candidate set: possible actions
- constraints: objectives (feasibility conditions)
- outcome: admissible action or action set

Decision-making is therefore formulated as a reduction of the feasible set rather than as a search trajectory.

Execution Layer (Robotics / Control)

At the execution level, selected actions are realized through physical or simulated processes. This layer typically operates over continuous state and control spaces and relies on planning, control, or optimization methods.

In this layer:

- input: selected action
- mechanism: trajectory planning, control, feedback
- outcome: physical realization

Unlike the upper layers, execution generally requires iterative computation due to continuous dynamics and environmental uncertainty.

Unified Interpretation

These three layers form a coherent structure:

- Perception eliminates representations
- Decision eliminates actions
- Execution realizes the selected action

The first two layers share a common deterministic elimination mechanism, while the execution layer provides the means of physical realization. This separation clarifies that iterative optimization is not a fundamental requirement of autonomy itself, but rather a property of the execution process in continuous domains.

Structural Insight

This layered view suggests that autonomous behavior can be decomposed into:

- a structural inference process (perception + decision), and
- a dynamic realization process (execution).

Within this framework, intelligence emerges not from iterative search alone, but from the structured reduction of possibilities under constraints, followed by appropriate physical realization.

Perception identifies what is, decision selects what is admissible, and execution realizes what is selected.

Conclusion

This study introduced a deterministic framework for autonomous decision-making based on an open objective architecture and finite candidate evaluation. In contrast to prevailing approaches that rely on iterative learning, optimization, and parameter

updates, the proposed model formulates autonomy as a structural process in which decisions emerge through the elimination of infeasible alternatives.

The framework demonstrates that a wide class of decision problems can be represented using discrete candidate sets, deterministic objectives, and admissibility constraints. System evolution occurs not through convergence in a parameter space, but through monotonic reduction of feasible sets and transitions across structured states. The introduction of an admissibility rule ensures that objective growth remains consistent, allowing the system to operate under an extensible objective structure without loss of feasibility.

The extension to relational and multi-stage settings further shows that autonomous behavior can be interpreted as the navigation of interconnected feasible sets. Actions are not globally defined, but arise within context-dependent clusters, where shared and conditional structures determine availability. This enables the construction of action sequences without requiring global optimization or iterative search.

The conceptual examples illustrate that the same framework applies across different levels of capability. Whether an agent directly executes an action, requests assistance, or operates with limited internal resources, the underlying mechanism remains unchanged: admissible actions are selected based on structural compatibility within the current state.

From a computational perspective, the framework shifts the focus from trajectory-based search to structure-based elimination. The primary limitation is the compatibility of the objective set and the coherence of relational constraints.

This work does not aim to replace learning-based or optimization-based systems. Rather, it identifies a complementary formulation of autonomy in which decision-making is governed by explicit structure instead of iterative adaptation. Such a perspective may be particularly relevant in settings where candidate sets are finite, constraints are well-defined, and interpretability is essential. Many practical solutions observed in everyday life reflect structured decision processes based on available resources and constraints, rather than explicit optimization or learning. Here, an autonomous system is formalized that already exists in human practice. The proposed framework represents a form of deterministic decision-making with constraints, in which solutions emerge through the elimination of infeasible alternatives rather than through iterative feasible search. Similar patterns can be observed in everyday practice, where decisions are guided by available resources and constraints. The framework formalizes deterministic, decision-making with constraints and reflects patterns commonly observed in real-world practice, where solutions arise through the direct use of available resources under constraints.

This work does not aim to compete with learning-based or optimization-based systems in large-scale or data-intensive settings; rather, it isolates and formalizes a complementary deterministic regime in which decision-making is governed by structural feasibility and constraint compatibility rather than iterative improvement, training, or convergence.

Author Contributions

Huseyin Murat Cekirge is the sole author. The author read and approved the final manuscript.

Conflicts of Interest

The author declares no conflicts of interest.

References

1. Bishop CM (2006) Pattern Recognition and Machine Learning. New York, NY, USA: Springer https://link.springer.com/book/9780387310732.
2. Goodfellow I, Bengio Y, Courville A (2016) Deep Learning. Cambridge, MA, USA: MIT Press https://www.deeplearningbook.org/.
3. Sutton RS, Barto AG (2018) Reinforcement Learning: An Introduction, 2nd ed. Cambridge, MA, USA: MIT Press https://psycnet.apa.org/record/2019-19679-000.
4. Murphy KP (2012) Machine Learning: A Probabilistic Perspective. Cambridge, MA, USA: MIT Press https://probml.github.io/pml-book/book0.html.
5. Szeliski R (2010) Computer Vision: Algorithms and Applications. London, UK: Springer https://link.springer.com/book/10.1007/978-3-030-

34372-9.

6. Jain AK, Duin RPW, Mao J (2000) "Statistical pattern recognition: A review," IEEE Transactions on Pattern Analysis and Machine Intelligence 22: 4-37.