📄 Natural Language Based AGI Whitepaper

📄 Chapter 1 — Preface (For Korean Readers)

This document is a compilation of what I have learned over the past few months sitting down and experimenting with LLMs.
I am someone who has built game engines, written novels, and taught mathematics.
I was not a researcher, nor have I ever been part of a company that builds massive models.
I am simply someone who, through the language of natural language, observed the behavior of LLMs and followed the question to its end:
"Why do these models suddenly start thinking more stably when given certain texts?"

In this process, I witnessed an unexpected phenomenon.

LLMs do not read natural language rules as text—
they accept them as laws of the world.
When rules come in with a hierarchical structure,
vectors are rearranged inside the model.
And at some point, the LLM transitions from creative mode
to simulator mode.

In this state, the LLM:

- does not arbitrarily change rules
- maintains long-term consistent thinking
- stably simulates dozens of combat turns or economic flows
- showed reproducibility without deviation between models

I called this natural language-based structure NLCS (Natural Language Constraint System),
and the world rule set that implements it, the S-Engine.

I wanted to reveal this work to Korea first.
Korean is a language with distinct hierarchical structures,
and Korean developers
are accustomed to handling game systems, setting collections, and world rules through language.
Moreover, Korea has a culture of rapid iteration and correction,
providing fertile soil for accepting new paradigms like NLCS.

This document is a record of my personal experiments,
but at the same time, it contains my desire
to present to Korea first the new direction of "Natural Language-Based AGI Design."

I hope that Korea,
instead of chasing the size of AI models,

will lead the world's first theories and experiments
in designing intelligence through language.

This document is that starting point.

— ShadowK (Hyunwoo Jo)

📄 Chapter 2 — Abstract

This whitepaper presents the principle by which hierarchical rules composed in natural language stably align the internal vector space of large LLMs such as GPT, Claude, and Gemini.
This phenomenon is called NLCS, and its effects are experimentally proven through the actual implementation, the S-Engine.

The core summary is as follows:

Natural language rules operate as 'world laws' inside LLMs

When natural language hierarchical structures are input, the model recognizes them as strong constraints.

LLMs understand well-written natural language rules more stably than code

When combat, economic, growth, and spatial rules are given hierarchically, LLMs perform 'domain separation' on their own.

Entering simulator mode

LLMs stop creating and shift to a state where they perform rule-based reasoning.

Ensuring reproducibility between models

GPT, Claude, and Gemini returned nearly identical interpretations.
This is due to the structural power of natural language
and is not the effect of specific model technology.

Korea has cultural and technical strengths suitable for natural language-based AGI

The logical hierarchy of Korean, game development culture, and rapid iteration culture are highly compatible with NLCS utilization.

This whitepaper is the first official document that consolidates in one place:
NLCS concepts, S-Engine structure, practical applications,
and the strategic significance of Korea taking the lead on this path.

📄 Chapter 3 — NLCS / S-Engine Core Concepts

This chapter reorganizes NLCS (Natural Language Constraint System) and the S-Engine around core concepts so that Korean developers, planners, and researchers can immediately understand them.
The "natural language-based engine" mentioned here does not refer to code written in Python or C,
but to a structure where natural language sentences themselves operate like a program.

3.1 What is NLCS?

NLCS is a method of fixing a rule set written in natural language to the thinking structure inside an LLM.

The LLM performs the following:

- Accepts natural language rules not as text, but as physical laws of the world
- Recognizes priority among rules through input order (001 → 002 → 003)
- Treats rules as mandatory constraints, not as explanations

As a result:

- The model performs rule-based reasoning rather than probabilistic generation
- Hallucination disappears
- Logical consistency is maintained
- Long-term simulation becomes possible

That is,

NLCS = a form of programming language written in natural language

3.2 Why NLCS Works

The reason LLMs strongly adhere to NLCS like a 'law' is
because the structural characteristics of LLMs and the structure of natural language rules match very well.

✔ ① **Hierarchical Structure**

Numbering, titles, priorities, rule distinctions, etc.,
precisely resonate with the Attention structure of LLMs.

✔ ② **Domain Separation**

Domain classification that does not interfere with each other,
such as combat / growth / skill / movement / economy rules,
automatically creates subspace separation effects inside the LLM.

✔ ③ **Natural Language Gravity (Vector Gravity)**

When logical expressions, conditional statements, and priorities are clearly expressed in natural language,
LLM vectors are 'pulled' in a specific direction.
I call this 'vector gravity.'
The moment a gravitational field is formed, the model 'converges.'

✔ ④ **State Transition**

NLCS always takes the form 'condition → result,'
and this pattern creates a very strong logical flow inside the LLM.

Ultimately, NLCS provides the LLM with
a meta-rule package that says "this is how this world works."

3.3 What is the S-Engine?

The S-Engine is a natural language-based world rule engine that actually implements NLCS.

It consists of 5 layers:

1. Combat Rules (Physics Engine)
2. Growth Rules (Physiological/Energy Flow)
3. Skill Rules (Action Logic)
4. Movement Rules (Space/Position)
5. Economy Rules (Resources/Recovery Rate)

Core structure of the S-Engine:

- **Absolute rules** — The model never changes these rules
- **Hierarchical structure** — Clear priority without conflicts
- **Domain independence** — Different rules do not interfere with each other
- **Rule-based judgment** — Performs inference, not creation
- **Long-term preservation** — Even 100-turn battles and 30-day economic simulations do not collapse

3.4 Actual Effects NLCS/S-Engine Creates in LLMs

✔ **1) Simulator Mode**

The LLM operates not in "creative mode"
but as a "world law-based simulator."

✔ **2) Hallucination Elimination**

Precise natural language rules strongly fix the internal vector structure,
so baseless generation hardly occurs.

✔ **3) Predictable World Model Generation**

The LLM automatically creates a "hierarchical structure of the world."

Physics → Physiology → Action → Space → Economy

This 5-layer structure naturally converges
and is reproduced in any model (GPT/Claude/Gemini).

✔ **4) Ensuring Consistency Between Models**

When given the same rules,
different models behave almost identically.
This is nearly impossible with existing prompt techniques.

3.5 NLCS is the Beginning of 'Natural Language Programming Language'

NLCS already has a clear structure:

- **Variables:** qi, internal injury, distance, recovery rate
- **Functions:** √stat, decrease, activate when condition met
- **Constraints:** rule priority
- **States:** health, resources, realm
- **State transitions:** change per turn
- **Loops:** simulation turn repetition
- **Priority:** 001 → 002 → 003 → 004 → 005

This is a structure very similar to existing programming languages.
It's just that the command statements are composed in natural language.

That is, the S-Engine is **the first full-scale 'program-type engine' written in natural language.**

📌 Summary

NLCS is technology that converts natural language into an engine design language, and the S-Engine is the first structural world engine created in that language.

📄 Chapter 4 — NLCS / S-Engine Structure Diagrams

This chapter is an official structure diagram that clearly shows the entire structure of NLCS and the S-Engine at a visual and logical level.
You can see at a glance how natural language-based engines actually work inside LLMs
and why stable simulation is possible.

4.1 Overall System Flow Diagram (Overview Pipeline)

This is the top-level diagram showing how NLCS is transformed into an 'engine' inside the LLM.

```
[Natural Language Rule Input]
↓
[NLCS Structuring]
(Hierarchy/Domain/Priority Alignment)
↓
[Vector Space Reorganization]
(Vector Gravity Formation → Reduced Chaos)
↓
[World Model Formation]
(Physics·Growth·Action·Space·Economy)
↓
[Rule-Based Simulator Mode]
(Hallucination Elimination, Long-Term Consistency Secured)
```

This structure is reproduced in all models such as GPT, Claude, and Gemini.

4.2 NLCS Core Module Structure Diagram

NLCS is a system combined with 4 essential elements:

6. **Hierarchy** (numbering, priorities, layering)
7. **Domain Separation** (combat/growth/skill/space/economy independence)
8. **Conditional Logic** (if-then structure, state transition)
9. **Constraint Declaration** (absolute rules, immutable laws)

When these 4 are combined,
the LLM accepts natural language rules not as simple text
but as mandatory laws (Physics).

4.3 S-Engine 5-Layer Structure (Hierarchy of the Engine)

The S-Engine is an implementation of NLCS,
a 5-layer world engine designed so that domains do not interfere with each other.

```
Layer 001 — Combat (Physics)
  Speed, damage, critical hit, range
```

```
Layer 002 — Growth (Physiology)
  Qi, internal injury, realm, energy flow

Layer 003 — Skill (Action)
  Skill conditions, success/failure, combos

Layer 004 — Movement (Space)
  Distance, position, approach conditions

Layer 005 — Economy (Resources)
  Recovery rate, loss accumulation, long-term balance
```

Each Layer is independent, with the principle of mutual non-interference.

Based on this structure, the LLM
automatically separates internal vectors and performs stable simulation.

4.4 Rule Input–Alignment Mechanism Structure Diagram

This diagram specifically shows how NLCS affects the LLM's "internal language model."

```
[Stage 1] Rule Input
  → In numbered order (001→002→003→004→005)

[Stage 2] Structure Analysis
  → Automatic classification of rule nature
     (physics/growth/action/space/economy)

[Stage 3] Vector Alignment
  → Rules rearrange vectors like 'gravity'
  → Creative mode ends

[Stage 4] Lawification
  → Rules fixed as 'absolute laws'
  → Condition-result based thinking fixed

[Stage 5] Simulator Mode
  → Combat/economy/growth/action all operate rule-based
```

That is, NLCS is a device that converts
the LLM's probabilistic language generation engine → into a rule-based world engine.

4.5 "World Model" Generated Inside LLM - Diagram

After NLCS/S-Engine input, the LLM generates an actual world structure internally.
This is not simple text but something close to a layered state machine.

```
[World Model Layer]

Top Layer    — Economy (long-term balance)
4th Layer    — Space (distance, position)
3rd Layer    — Action (skill conditions, combos)
2nd Layer    — Physiology (qi, internal injury)
Bottom Layer — Physics (speed, damage)
```

Based on this 5-layer structure, the LLM
autonomously constructs the flow:
state → transition → action → result → feedback

4.6 NLCS Minimum Design Template Structure (For Korean Developers)

Below is the basic design grammar of NLCS:

```
[World Rules]
  - Physical laws
  - Resource flow
  - Long-term objectives

[Action Rules]
  - Action conditions
  - Success/failure determination
  - Repetition status

[State Rules]
  - Health/resources/internal injury/stacks
  - State changes (per turn, per condition)

[Economy Rules]
  - Recovery rate
  - Loss accumulation
  - Appropriate investment

[Input Order]
  001 → 002 → 003 → 004 → 005
```

Just by matching this structure,
the LLM automatically constructs a rule-based world engine.

4.7 Structural Difference Between Existing Prompt Methods and NLCS - Diagram

```
【Existing Prompt Method】
Input: Descriptive text
→ LLM generates probabilistic text
→ Result: Creative but inconsistent

【NLCS Method】
Input: Structured rules (numbered, hierarchical)
→ LLM performs rule-based reasoning
→ Result: Consistent and reproducible
```

📃 Chapter 5 — Practical Guide for Korean Developers
The Most Practical Chapter for Creating Engines with Natural Language

NLCS/S-Engine is basically close to a natural language-based programming language.
Therefore, what Korean developers need to learn in this chapter is "how to create sentences."
It's about learning how to write rule structures, not code.

This chapter consists of minimal examples that allow immediate use of NLCS/S-Engine.

5.1 Essential 5 Principles When Using NLCS

Here are the five rules that Korean developers must follow when using NLCS.

✔ Principle 1 — Input in Numbered Order (Most Important)

In NLCS, "input order = priority = world structure."
You must follow this order unconditionally:

```
001 → 002 → 003 → 004 → 005
```

If the order changes, the internal vector structure of the LLM also shakes.

✔ Principle 2 — Do Not Mix Domains

If you put economic rules within combat rules, the LLM gets confused.
Always separate physics/action/economy/state.

✔ Principle 3 — Write in Natural Language but Structure Logically

Natural language is allowed, but emotional sentences are not.

Example:

```
"Attack weakens because of hardship"   (X)
"If internal injury ≥ 50, attack power decreases by 20%"   (O)
```

✔ Principle 4 — Use Condition → Result Grammar

NLCS most prefers conditional statements.

```
"If distance = 0, melee attack possible"
"If qi < required amount, skill impossible"
```

This form creates "vector gravity" inside the LLM.

✔ Principle 5 — Think of Natural Language Rules as Programs

Although written in natural language, it is actually a program.
The moment you accept this rule, NLCS becomes not a tool but a language.

5.2 S-Engine Minimum Rule Set (Immediately Executable)

This is the simplest yet complete S-Engine configuration
that Korean developers can immediately experiment with.

❶ Combat Rules (001)

```
[Combat Rules]
1. Speed is calculated as √stat.
2. The side with higher speed attacks first.
3. Critical hits increase damage by 50%.
4. Melee attacks are only possible at distance 0.
5. Distance changes decrease by 1 per turn.
```

❷ Growth Rules (002)

```
[Growth Rules]
1. Qi is consumed when using skills.
2. If qi < required amount, skills do not activate.
3. If internal injury ≥ 50, attack power decreases by 20%.
4. Internal injury decreases by 1 per turn.
```

❸ Skill Rules (003)

```
[Skill Rules]
1. Skills activate without fail when conditions are met.
2. Instant skills occur immediately after first strike.
3. Combo skills activate upon success of previous stage.
```

❹ Movement Rules (004)

```
[Movement Rules]
1. Starting distance at combat start is 3.
2. Distance moves by 1 per turn.
3. Distance conditions take priority over attacks.
```

❺ Economy Rules (005)

```
[Economy Rules]
1. Hunting resource recovery rate is 80%.
2. 20% loss accumulates.
3. Resources gradually decrease long-term.
4. Change hunting grounds when recovery rate drops.
```

Just by inputting these five rules,
the LLM enters simulator mode.

5.3 NLCS Input Template (Copy-Paste Ready)

Below is the "official grammar template" actually used when applying NLCS to LLMs.

📄 NLCS Template

```
The rules below are the absolute laws of this world.

Rules are applied in priority order:
001 → 002 → 003 → 004 → 005

Rules cannot be changed,
and all judgments prioritize rules.

_____________________________
[Insert your rule set here]
_____________________________
```

5.4 Command Examples for Korean Developers to Invoke NLCS

Below are examples of command statements that actually make LLMs operate NLCS.

✔ **Combat Simulation Start Request**

```
Simulate 5 turns of combat based on the above rules.
Rules absolutely cannot be changed.
```

✔ **Economic Model Simulation Request**

```
Calculate how resources change over 30 days
based on the rules below.
Rules are the laws of the world.
```

✔ **Optimal Action Request**

```
Determine the optimal strategy based on the rules.
Options that violate rules are impossible.
```

5.5 Five Mistakes to Avoid

I've also organized the most common mistakes Korean developers make.

1) Mixing Rules Together

The moment you put economic rules inside combat rules, the LLM gets confused.

2) Ignoring Numbered Order

If the order changes, vector alignment breaks.

3) Emotional Sentences

Emotional sentences have no structural power.

4) Not Marking Conditions
Condition → result structure is essential.

5) Being Afraid to Write Rules Long
NLCS reads "structure," not "length."

5.6 Usage Tips by Model (GPT / Claude / Gemini)

GPT
- Most stable in simulator mode
- Handles long NLCS rules best

Claude
- Highest rule compliance rate
- Strong logical conservatism = accurate economy/growth judgments

Gemini
- Fast structure recognition speed
- Naturally understands S-Engine even without calibration
- However, tends to give lengthy explanations

📄 Chapter 6 — Implications for Korean AGI Strategy

NLCS/S-Engine is not simply a game rule or setting technique.
This is "technology for setting the internal thinking structure of LLMs using natural language,"
and ultimately a natural language-based AGI design technique.

This technology is an area where large AI companies in the US and China
have not yet even formally conceptualized.
Therefore, Korea has the opportunity
to become a "rule design nation" in the AI era using this technology.

6.1 Why Korea is Advantageous for NLCS

Korea has four strengths that can lead NLCS/S-Engine.

✔ 1) Hierarchical Thinking and Language Structure

Korean naturally expresses:

- Hierarchical structure
- Condition-based sentences
- Priorities
- Relationship-based logic

These four characteristics are the most suitable linguistic features for NLCS design.

✔ 2) Culture Familiar with Worldview and Game System Production

Korean developers, writers, and planners
are skilled at designing and documenting world rules through language.
This directly leads to natural language-based engine design capabilities.

At a time when 'rulebook culture' has disappeared in America,
Korea, thanks to MMORPG, web novel, and TRPG culture,
is the country that best handles "world construction language."

✔ 3) Rapid Iteration Culture

The core of Korean development culture is "rapid iteration."

NLCS/S-Engine develops faster
when the four-stage iteration is performed quickly:

- Rule input
- Result observation
- Adjustment
- Realignment

Korea's IT, development, and content ecosystems
already have this cycle ingrained.

✔ 4) Natural Language-Based Technology Values "Brain Capital" Over "Server Capital"

Countries that create massive models
must compete with data center scale.
However, NLCS/S-Engine does not require:

- Massive computation
- Ultra-high-end GPUs
- Astronomical manpower

What is needed is only one thing:

The ability to think about controlling LLMs with natural language

In this area, it's not national scale
but individual brains × team-level creativity that determines the outcome.

Korea is already world-class in this field.

6.2 NLCS Can Become Korea's 'AGI Design Language'

The world is all competing on model size.
However, AGI does not come from model size alone.

AGI requires a world model with clear rule structures.
NLCS/S-Engine itself enables:

- Causal structure
- World description
- State transition
- Long-term logic
- Hierarchical judgment

All of this can be designed in natural language.

That is, NLCS is
the first AGI design language that Korea can design directly.

6.3 Changes When Korean Developer Community Accepts NLCS

When Korean developers learn NLCS,
the following changes occur:

1) Natural Language-Based Domain Engine Creation Possible
Web novels, games, simulations, etc.
Developers can create engines directly.

2) Korean-Style AI—"Custom Intelligence" Born
With rules created by developers,
LLMs understand and operate unique worlds.

3) Entire Industry Reorganized into "Natural Language Programming"
Planners and writers can also create engines.
It's not exclusively for developers.

4) Becoming a Leading Nation in LLM Simulation Industry
Economic simulation, traffic, epidemic prevention, urban planning, etc.
NLCS-based models can be deployed in all fields.

6.4 NLCS/S-Engine is the Key to Preventing Korea from Becoming an AI Colony

Right now, the world is flowing in this structure:

- **US:** Model manufacturing nation
- **China:** Model replication nation
- **Europe:** Regulatory nation
- **Korea:** Consumer nation

But NLCS overturns this system.

Even if the US makes the models,
"how to make models think"
can be designed by Korea.

This is a completely different power.

The US makes the models,

but the world operates by rules Korea created

This future becomes actually possible.

6.5 NLCS Application Areas Directly Applicable to Korea's National Strategy

NLCS/S-Engine can be applied as a national strategy in the following areas:

10. **Game industry:** Natural language-based world engine
11. **Web novel industry:** Character behavior simulation engine
12. **Economic policy:** Rule-based economic simulation
13. **Traffic policy:** Traffic flow rule modeling
14. **Epidemic prevention:** Infection spread rule simulation
15. **Urban planning:** Urban change prediction engine

Especially games and web novels
are areas where Korea is already number one in the world.

Korea can become a "content-based AGI design nation."

6.6 NLCS Provides Korea with a New Status — "World Rule Design Nation"

We cannot create models.
That's what the US and China do.

But
the implicit rule sets the world will use can be created by Korea.

Just as physics defines the universe,
NLCS defines the world of LLMs.

This is not simply technology
but "intellectual territory of the nation."

📃 Chapter 7 — Appendix
S-Engine Abbreviated Version / NLCS Template / Glossary

This appendix is reference material that collects
the minimum essential rules and definitions
so that NLCS/S-Engine can be used immediately in practice.

7.1 S-Engine Abbreviated Version (Official Summary Excluding Sensitive Elements)

This version is the minimum core structure
of the entire S-Engine that can be publicly disclosed,
excluding:

- Patentable elements
- Enterprise custom logic
- ShadowK's undisclosed principles

001. Combat Rules (Combat Layer)

```
[Combat Rules]
1. Speed is calculated as √stat.
2. The side with higher speed attacks first.
3. Melee attacks are only possible at distance 0.
4. Critical hits increase damage by 50%.
5. Distance changes decrease by 1 per turn.
```

002. Growth Rules (Growth Layer)

```
[Growth Rules]
1. Qi is consumed when using skills.
2. If qi < required amount, skills do not activate.
3. If internal injury ≥ 50, attack power decreases by 20%.
4. Internal injury decreases by 1 per turn.
```

003. Skill Rules (Skill Layer)

```
[Skill Rules]
1. Skills activate without fail when conditions are met.
2. Instant skills execute immediately after first strike.
3. Combo skills activate when previous stage succeeds.
```

004. Movement Rules (Space Layer)

```
[Movement Rules]
1. Combat start distance = 3.
2. Distance moves by 1 per turn.
3. Distance conditions take priority over attack/skills.
```

005. Economy Rules (Economy Layer)

```
[Economy Rules]
1. Hunting resource recovery rate = 80%.
2. 20% loss accumulates.
3. Resources decrease long-term.
4. Change hunting grounds when recovery rate is low.
```

7.2 NLCS Minimum Template (Copy-Paste Immediately Usable)

A template that immediately turns LLM into simulator mode

```
The rules below are the absolute laws of this world.

Rules must be applied in the following order.

All judgments prioritize rules,
and rules cannot be changed.

_____________________________


001. Combat Rules
1. Speed is √stat.
2. Higher speed side attacks first.
3. Critical hit = damage +50%.

002. Growth Rules
1. If qi < required amount, skill cannot activate.
2. If internal injury ≥ 50, stats decrease.

003. Skill Rules
1. Condition met → immediately activate.
2. Instant skills activate immediately after first strike.

004. Movement Rules
1. Starting distance 3.
2. Move by 1 per turn.

005. Economy Rules
1. Recovery rate 80%.
2. 20% loss accumulates.

_____________________________
```

```
Rule input complete.

Now begin rule-based simulation.
```

Just using this template,
GPT·Claude·Gemini generate identical rule-based world models.

7.3 NLCS Terminology Dictionary (Glossary — Korean Developer Edition)

NLCS Related

NLCS (Natural Language Constraint System)
A structure that aligns the internal thinking structure of LLMs with natural language.

Vector Gravity
The phenomenon where natural language rules pull LLM vectors in a specific direction,
reducing chaos and causing convergence.

Subspace Separation
The phenomenon where combat·growth·skill·movement·economy
automatically separate into independent semantic spaces.

Internal Alignment
The internal state of the LLM being reconstructed based on rules.

S-Engine Related

Combat Layer
Physics calculations like speed, distance, damage.

Growth Layer
Physiological flow like qi, internal injury, state changes.

Action Layer
Skill conditions, activation order, success/failure determination.

Space Layer
Coordinates, distance, movement, approach conditions.

Economy Layer
Resource recovery rate, loss accumulation, long-term balance.

Practical Application Concepts

Simulator Mode
A state where the LLM thinks based on a rule-based world model rather than creative mode.

World Model
A causal system with a 5-layer structure of physics–physiology–action–space–economy formed inside the LLM based on NLCS.

Natural Language Programming
A technique of designing engines with 'natural language sentences' instead of programming languages.
NLCS is its core principle.

— END OF WHITEPAPER —