Natural Language → Addressable State
A Pre-Execution Translation Layer for State-Conditioned AI Systems

DHEX STL

DX (Yeonseok Jeong)

Creator of DesignHEX
and the DHEX STL architecture

Preprint v1.0
2026

Abstract

Contemporary AI systems operate through probabilistic interpretation of natural language,
yet lack a formal mechanism to fix the condition under which execution is permitted.

We define this structural absence as the Decision Gap.

This paper introduces the DesignHEX State Translation Layer (DHEX STL),
a pre-execution architecture that transforms natural language into a
deterministically addressable state within a constrained state space.

Formally:

STL: $\mathscr{L} \rightarrow \mathscr{H}$

where $\mathscr{L}$ denotes the natural language space,
and $\mathscr{H} \subset \mathbb{Z}_{16}^{6}$ represents a bounded, discrete state space.

DHEX STL is not an interpretive function,
but a state fixation operator enabling execution
to be conditioned on a non-inferential, pre-declared state.

This enables a structural shift from probabilistic inference
to state-conditioned execution.

Keywords:
Pre-Execution Architecture, State Translation, Decision Architecture, AI Governance,
Addressable State, Natural Language Processing, State-Conditioned Systems,
Non-Executable Systems, Decision Gap, AI Control Layer

The Decision Gap

Current AI systems can be expressed as:

$y = f(L)$
$f: \mathscr{L} \to \mathscr{Y}$

where L denotes natural language input and y denotes output.

This structure exhibits:

- non-determinism
- contextual dependency
- post-hoc validation

This results in the following condition:

$\exists\, y \in \mathscr{Y}$ such that y is executable $\wedge$ y is retrospectively invalid

We define this structural inconsistency as the Decision Gap:

Execution occurs without a fixed decision condition.

From Interpretation to Fixation

Traditional AI pipeline:

Natural Language → Inference → Output

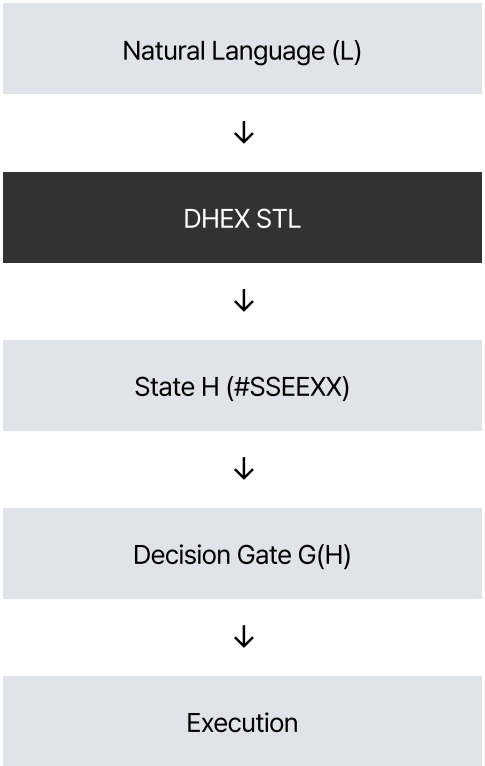
This model relies on interpretation,
which introduces variability and ambiguity.

We propose a structural shift:

Interpretation → Translation → Fixation

Natural language must not be interpreted,
but transformed into a fixed state prior to execution.

[Figure 1. Pre-Execution State Translation Pipeline]



Natural language is translated into a fixed state prior to execution.
Execution is conditioned on H, not inferred from L.

STL Formalization

We define the DesignHEX State Translation Layer as a composite operator:

STL ≔ $\mathscr{E} \circ \mathscr{N} \circ \mathscr{C}$

$\mathscr{C}$: $\mathscr{L} \to \mathscr{I}$ (Extraction)
$\mathscr{N}$: $\mathscr{I} \to \mathscr{S}$ (Normalization)
$\mathscr{E}$: $\mathscr{S} \to \mathscr{H}$ (Encoding)

Thus:

STL: $\mathscr{L} \to \mathscr{H}$

This formulation describes a structured transformation
from language space into a discrete state space.

State Space Definition

We formally define the state space as:

$\mathscr{H} = \{ (S, E, X) \in (\mathbb{Z}_{16}^2)^3 \}$

$|\mathscr{H}| = (16^2)^3 = 16^6 = 16{,}777{,}216$

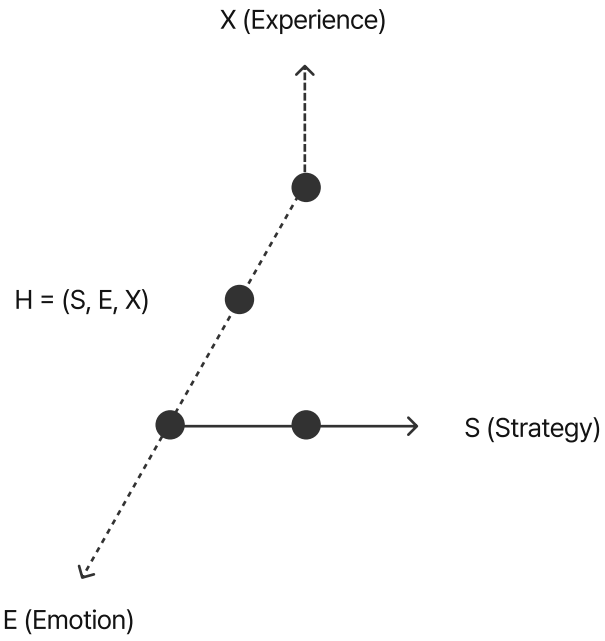
Each state is represented as:

H ≡ #SSEEXX

This defines a bounded, discrete state lattice with:

- addressability
- finiteness
- non-continuity

[Figure 2. Discrete State Space Representation (S, E, X Axes)]



Non-Executable Constraint

We impose the following condition:

¬∃ algorithm A such that
A(L) = H for all L ∈ $\mathscr{L}$

This implies that the mapping from natural language to state
is intentionally non-computable in the general case.

This constraint ensures:

- irreducibility
- non-replicability
- controlled interpretability

Layered Translation Structure

The STL operates through three layers:

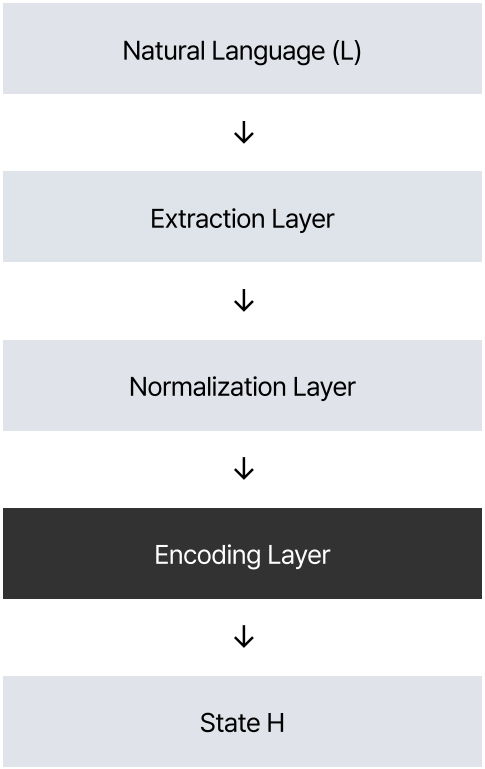
1. Extraction
$L \rightarrow (i_1, i_2, \ldots, i_n)$

2. Normalization
$(i_1 \ldots i_n) \rightarrow (S, E, X)$

3. Encoding
$(S, E, X) \rightarrow H$

Each layer reduces ambiguity and increases structural determinism, culminating in a fixed state representation.

[Figure 3. Internal Structure of the DHEX State Translation Layer]



Extraction: semantic components (intent, context, risk)
Normalization: structural alignment into S, E, X axes
Encoding: transformation into discrete addressable state

Each layer reduces ambiguity and increases structural determinism.

State-Conditioned Decision

We define a decision function:

$G: \mathscr{H} \to \mathscr{D}$

$\mathscr{D}$ = {ALLOW, HOLD, REJECT}

With a validation predicate V:

G(H) =
ALLOW if V(H)
HOLD if ¬V(H) ∧ uncertainty(H) > τ
REJECT otherwise

where τ denotes a predefined uncertainty threshold.
All decisions are derived from state H, not from natural language.

Minimal Translation Case

Input:
“Can this image be used in an advertisement?”

Translated State:
H = #A3F29C

Condition:
V(H) = false

Decision:
G(H) = HOLD

The decision is not inferred during execution.
It is already encoded in the state.

System Implications

AI Systems:
Transition to state-conditioned execution

Governance:
Pre-execution accountability and traceability

Product Systems:
Addressable decision interfaces and state referencing

This enables a structural separation between
generation and admissibility.

Disclosure Boundary

The following components are intentionally not disclosed:

- mapping functions
- encoding logic
- boundary definitions
- validation conditions

These elements are governed by the
Non-Executable State Evaluation Principle.

Conclusion

We define a new structural pipeline:

$\mathscr{L} \rightarrow \mathscr{H} \rightarrow G(H)$

AI systems have solved execution.

What remains unresolved is the condition
under which execution should exist.

DHEX STL introduces a pre-execution layer
that defines admissibility before action.

DX (Yeonseok Jeong)

Creator of DesignHEX
and the DHEX STL architecture

Preprint v1.0
2026