Redefining Semantics in the Age of AI: From Static Meaning to Evolving Constraints

Y. Matsuda

February 12, 2026

Abstract

The concept of *semantics* has historically been defined as a mapping from *syntax* to meaning. In compiler theory, semantics assigns syntactic structures to well-defined semantic objects. In formal semantic theory, meaning is grounded in truth conditions within structured models. Both approaches presuppose closed domains and stable interpretation spaces.

The emergence of Large Language Models (LLMs) and agent architectures challenges this foundation. In LLMs, *context* no longer functions as an auxiliary parameter but as a probabilistic constraint field over high-dimensional state space. Meaning becomes conditioned transition rather than static denotation. In agent systems operating in open environments, context further evolves into an action-conditioned structure that continuously reshapes constraints. Here, *constraint* is not merely a rule to be satisfied; it becomes an operator governing admissible state evolution.

This paper proposes a redefinition:

> Semantics in the age of AI is the coupled evolution of state transitions and constraint structures under contextual deformation.

We examine this transformation across four stages: (1) classical compiler semantics as syntax-to-object mapping, (2) formal semantic theory as truth-conditional modeling, (3) LLM-based systems where context induces probabilistic constraint stabilization, and (4) agent architectures where constraint update becomes integral to action in open worlds. We introduce a geometric interpretation in which context acts as a local coordinate system over state space, and we formalize constraint update as an operator on that space.

Finally, we argue that this redefinition carries architectural implications. If semantics is constraint-stabilized and constraint-evolving transition, then AI systems require explicit constraint governance. Constraint-Driven Development (CDD) is introduced as the engineering manifestation of this semantic shift, serving as a bridge between philosophical reinterpretation and operational system design.

This work repositions semantics from static meaning to evolving constraints, framing meaning as the governed evolution of state transitions in open, context-dependent AI systems.

1 Introduction

The word *semantics* traditionally refers to meaning. In computer science, semantics was formalized through precise correspondences between syntactic structures and mathematical objects. In artificial intelligence, however, systems now operate through large-scale probabilistic inference and interaction with open environments. Under these conditions, the classical understanding of semantics as a static mapping from *syntax* to meaning becomes insufficient.

This paper proposes a conceptual reorganization: semantics in the age of AI should be understood as

state transitions that remain stable under evolving constraints.

This shift preserves operational rigor while acknowledging the dynamical and contextual nature of AI systems.

We examine four historical stages:

1. Compiler semantics
2. Formal semantic theory
3. LLM-based contextual modeling
4. Agent-based open-world interaction

We show how each stage progressively moves semantics from static denotation toward dynamic constraint evolution.

2 Classical Compiler Semantics: Syntax-to-Object Mapping

In compiler construction, semantics is defined via explicit mappings from syntactic structures to semantic domains. Given a grammar G defining *syntax*, semantic rules define a function:

$$\mathcal{S} : \text{Syntax} \to \text{Semantic Domain}.$$

The semantic domain may consist of:

- operational behaviors,
- denotational mathematical objects,
- generated target code.

Meaning here is deterministic and compositional. Context is limited to lexical scope and well-defined environment frames. The system is closed. All semantic effects are predictable within the formal model.

This conception of semantics is precise but brittle. It assumes:

- fixed interpretation space,
- closed world,
- fully enumerated rules.

3 Formal Semantic Theory: Truth in Models

Formal semantic theory extended mapping-based semantics into natural language. Meaning became a function from expressions to truth conditions within models.

$$\text{Expression} + \text{Model} \to \text{Truth Value}.$$

While powerful, this framework presupposes:

- stable models,
- bounded interpretation domains,
- complete specification of context.

Natural language, however, introduces:

- indexicality,
- ambiguity,
- dynamic reference,
- incomplete world knowledge.

Attempts to fully formalize natural language semantics at scale encountered combinatorial explosion. Context was treated as an auxiliary parameter rather than a primary structural element.

4 LLMs: Context as Probabilistic Constraint Field

Large Language Models introduce a radical shift.

Tokens serve as *syntax*. However, meaning is not assigned via explicit mappings. Instead, each token sequence induces a high-dimensional state representation.

The next-token distribution can be described abstractly as:

$$P(t_{n+1} \mid t_1, \ldots, t_n).$$

Here, *context* is not a static parameter. It is the accumulated transformation of internal state.

We propose:

In LLMs, context functions as a probabilistic constraint field over the state space.

Each new token does not simply add content. It reshapes the permissible region of subsequent transitions. Semantics becomes:

> constraint-induced stabilization of token transitions.

Meaning is neither denotation nor truth. It is the narrowing of possibility space under contextual pressure.

Importantly: semantics here is not merely constraint satisfaction. It is constraint-conditioned transition.

5 Agents: From Closed Logic to Open Action

Agents extend LLMs by incorporating:

- tool invocation,
- memory updates,
- environmental feedback,
- world interaction.

This introduces a fundamental shift: the system becomes open.

Unlike logic programming systems such as *Prolog*, which operate in closed logical universes, agents function in open worlds.

In closed logical systems:

$$\text{Semantics} = \text{Model Satisfaction}.$$

In agents:

$$\text{Semantics} = \text{Action-Conditioned State Transformation}.$$

Constraints are no longer purely logical. They include:

- environmental affordances,
- tool availability,
- temporal continuity,
- world consistency.

Thus, constraints evolve through action.

We therefore refine our thesis:

> Semantics in agent systems is constraint-updating state transition.

Agents are both:

- generators of meaning (producing state trajectories),
- regulators of constraints (reshaping permissible futures).

6 From Constraint Satisfaction to Constraint Update

Is semantics equivalent to constraint satisfaction?

We argue: No.

Constraint satisfaction presupposes:

- fixed constraint sets,
- stable solution spaces.

AI systems continuously modify:

- internal states,
- contextual embeddings,
- memory structures,
- environmental states.

Thus, meaning must include constraint transformation.

We propose the following definition:

> **Definition.** Semantics is the dynamic stabilization of state transitions under evolving contextual constraints.

This definition accommodates:

- probabilistic modeling,
- tool-augmented reasoning,
- world-model integration,
- self-modifying agents.

7 Two Forms of Context

We distinguish:

7.1 LLM Context

- Accumulated token history
- Internal latent state
- Probabilistic constraint field

This context shapes distribution over next transitions.

7.2 Agent Context

- Interaction history
- External memory
- Tool results
- Environmental state

Here, context becomes action-conditioned constraint structure.
Thus, *context* is no longer auxiliary. It is the primary carrier of semantics.

8 World Models and Open Dynamics

Modern AI systems increasingly incorporate internal *world models*. Unlike classical semantic models, these are not static interpretive domains. They are predictive, generative, and counterfactual structures that simulate possible trajectories of interaction.

A world model can be understood as a structured hypothesis about how states evolve under actions:

$$\text{State}_t + \text{Action}_t \to \text{State}_{t+1}.$$

However, in open environments, neither the full state space nor the transition rules are fully known. The system operates under incomplete knowledge, uncertain dynamics, and evolving external conditions.

This creates a fundamental problem: how can semantics remain coherent in an open world?

8.1 Open Knowledge and the Grounding Problem

In open environments, knowledge is incomplete and continuously expanding. Agents encounter:

- previously unseen entities,
- novel combinations of actions,
- unexpected environmental feedback.

This condition may be described as *open knowledge*.

Open knowledge destabilizes classical semantic assumptions. There is no fixed model in which expressions are evaluated. Instead, meaning must remain stable despite partial information.

Here, *constraints* play a grounding role.

Constraints function as:

- boundary conditions on admissible states,

- invariants across evolving trajectories,
- coherence-preserving mechanisms.

They provide minimal structure sufficient to prevent semantic drift.

In this sense, constraints are not merely logical filters. They are grounding anchors in an otherwise fluid state space.

8.2 Constraint as Stability Operator

In an open world, transitions may diverge arbitrarily. Without regulation, state trajectories would exhibit semantic instability.

We may conceptualize a constraint as a stability operator: it restricts state transitions to a subspace where trajectories remain coherent.

This does not imply determinism. Rather, constraints define a basin of attraction within which transitions remain semantically meaningful.

Thus, semantics at the world-model level becomes:

> trajectory coherence under constraint stabilization.

8.3 From Logical Constraints to Action Constraints

In closed logical systems, constraints are declarative. They specify what must hold.

In open agent systems, constraints become operational. They regulate:

- action feasibility,
- temporal consistency,
- resource usage,
- interaction safety.

This marks a conceptual shift:

> Constraint evolves from a property of propositions to a regulator of action.

Agents no longer merely satisfy constraints. They collaborate with them.

8.4 Constraint as Collaborative Structure

An agent in an open environment continuously:

- proposes actions,
- evaluates outcomes,
- updates internal models,

- adjusts future possibilities.

Constraints participate in this loop.
They:

- shape proposal space,
- evaluate candidate transitions,
- update permissible regions,
- preserve structural integrity.

Thus, constraint is not external to the agent. It becomes a co-regulatory mechanism.
Semantics emerges from the interaction between:

- generative dynamics (agent proposals),
- stabilizing structures (constraints),
- environmental feedback.

8.5 World Models as Constraint-Conditioned Simulators

When agents employ internal world models for planning or simulation, they generate hypothetical trajectories.
Not all trajectories are semantically viable.
Constraints:

- prune incoherent simulations,
- eliminate unstable projections,
- bias planning toward stable attractors.

Meaning therefore resides not in isolated states, but in constraint-resilient trajectories.
This reframes semantics as:

> the persistence of structured behavior under uncertainty.

Open knowledge does not destroy semantics. It necessitates constraint-mediated stabilization.

8.6 Grounding Revisited

Grounding in AI has traditionally been framed as symbol-to-world correspondence. In open agent systems, grounding shifts:

- from symbol-to-object,
- to transition-to-outcome stability.

An action is meaningful if it produces constraint-consistent evolution.

Thus, grounding becomes dynamic.

Constraint serves as the minimal substrate that allows semantics to persist despite openness.

8.7 Context as Action-Conditioned Constraint Update

In earlier stages of semantic theory, *context* was treated as a parameter: a background against which expressions were interpreted. In LLM-based systems, context evolved into an accumulated latent state, functioning as a probabilistic constraint field over possible token transitions.

In agent systems, however, context undergoes a further transformation.

Context is no longer merely an accumulated history. It becomes a structure that is continuously modified through action.

Formally, we may describe this transformation schematically as:

$$\text{Context}_{t+1} = \mathcal{U}(\text{Context}_t, \text{Action}_t, \text{Environment Feedback}_t).$$

Here, context is dynamically updated. More importantly, it conditions how constraints themselves evolve.

Thus, context does not simply constrain transitions. It participates in constraint formation.

This yields a key thesis:

> Context becomes action-conditioned constraint update.

To unpack this claim, consider the following distinctions:

1. Context as Constraint Field (LLM Stage) In large language models:

- Context accumulates token history.
- It reshapes probability distributions.
- It restricts the admissible next-token region.

Constraints here are implicit and probabilistic. Context narrows transition possibilities but does not explicitly modify constraint structure.

2. Context as Constraint Modifier (Agent Stage) In agent architectures:

- Actions alter external states.
- External states feed back into memory.
- Memory updates reshape internal world models.

Constraints are therefore not static filters. They are updated as consequences of action. An action may:

- introduce new invariants,
- relax previous assumptions,
- invalidate prior trajectories,
- generate new feasible subspaces.

Context thus becomes a dynamic mediator between action and constraint evolution.

3. Constraint as Co-Agent In this formulation, constraint is not external to the agent. It acts as a structural counterpart.

The agent proposes transitions. The constraint structure:

- stabilizes,
- filters,
- reshapes,
- or expands

the admissible state space.

Meaning emerges from this interaction.

Semantics is no longer solely the interpretation of syntax under context. It is the co-evolution of:

- state trajectories,
- constraint structures,
- contextual representations.

4. Implications for Open Knowledge In open environments, context must integrate:

- newly acquired information,
- unexpected outcomes,
- revised world assumptions.

Without constraint update, context accumulation would lead to incoherence.
Therefore:

Context stabilizes semantics only insofar as it mediates constraint evolution.

This reframes context from passive background to active structural operator.
In the AI era, context is not the stage on which semantics plays out. It is the mechanism through which semantics reorganizes itself.

9 Operational Implications and Constraint-Driven Development

This section extends the operational perspective by introducing a structural interpretation of context and constraint evolution. We treat them not merely as design considerations, but as geometric and operator-level primitives in the semantic architecture of AI systems.

9.1 Context as Local Coordinate System in State Space

To further clarify the role of *context*, we introduce a geometric interpretation.
Let the internal dynamics of an AI system be defined over a high-dimensional state space $\mathcal{X}$. Each internal representation corresponds to a point $x \in \mathcal{X}$. State transitions are governed by a transition operator:

$$x_{t+1} = T(x_t, a_t),$$

where a_t may represent an action, token emission, or external interaction.
In classical semantics, interpretation assigns meaning by mapping syntactic objects into semantic domains. In LLMs and agents, however, meaning unfolds within trajectories in $\mathcal{X}$.
We propose that context functions as a *local coordinate system* over this state space.
Rather than defining meaning globally, context determines:

- which dimensions are salient,
- which directions are admissible,
- which regions are reachable,
- which trajectories are stable.

In this view, context does not merely store history. It reshapes the geometry of the state space locally.

A change in context corresponds to a deformation of local structure: certain transitions become amplified, others suppressed.

Thus, meaning is not located at a point. It resides in locally structured transition possibilities.

Semantics becomes:

> the organization of transition dynamics relative to a context-induced local geometry.

This interpretation explains why identical tokens may acquire different meanings under different contexts: the local coordinate system has changed.

In agent systems, context includes interaction history, environmental feedback, and memory updates. Therefore, the local geometry of the state space evolves with action.

Meaning is trajectory-dependent because the coordinate system itself is dynamic.

9.2 Constraint Update as an Operator on State Space

If context shapes local geometry, constraints regulate admissible transitions within that geometry.

We therefore formalize constraint update as an operator.

Let $\mathcal{C}_t$ denote the constraint structure at time t. Instead of treating constraints as static predicates, we define a constraint update operator:

$$\mathcal{C}_{t+1} = \Phi(\mathcal{C}_t, x_t, a_t, e_t),$$

where:

- x_t is the current internal state,
- a_t is the action performed,
- e_t is environmental feedback.

The operator Φ may:

- introduce new constraints,
- relax prior constraints,
- reinforce invariants,
- eliminate invalidated structures.

Constraints therefore evolve.

Meaning cannot be reduced to satisfying $\mathcal{C}_t$. It must include the dynamics of $\mathcal{C}_t$ itself.

This leads to a refined definition:

> Semantics is the coupled evolution of state transitions and constraint operators under contextual deformation.

In closed logical systems, $\mathcal{C}_t$ is fixed. In open agent systems, $\mathcal{C}_t$ becomes part of the dynamic state.

Thus, constraint is no longer merely a filter. It is an operator governing the admissible evolution of trajectories.

This perspective clarifies the distinction between:

- constraint satisfaction (static),
- constraint transformation (dynamic),
- constraint governance (architectural).

Agents operate in the second and third regimes.

They both generate trajectories and participate in constraint evolution.

If semantics is redefined as constraint-stabilized state transition, then system design must incorporate explicit constraint structures.

This perspective aligns closely with what may be termed *Constraint-Driven Development* (CDD).

CDD assumes:

- systems are shaped by structural invariants,
- correctness emerges from constraint preservation,
- flexibility arises from controlled constraint evolution.

9.3 Constraint as Architectural Primitive

Under the proposed semantic redefinition, constraints are not secondary validation layers. They are primary semantic carriers.

System architecture should therefore:

- represent constraints explicitly,
- monitor constraint stability across transitions,
- enable controlled constraint updates.

In agent systems, this includes:

- schema validation,
- action feasibility filters,
- world-model consistency checks,
- interaction safety boundaries.

9.4 Constraint Evolution as Semantic Maintenance

Because semantics includes constraint update, systems must track:

- when constraints are reinforced,
- when constraints are relaxed,
- when constraints are violated,
- when constraints must be reformulated.

CDD provides a structural approach to managing this evolution.

Rather than treating AI outputs as isolated artifacts, CDD interprets them as transitions within a constraint-governed space.

9.5 Semantic Integrity in Open Systems

In open environments, semantic failure manifests as:

- incoherent state transitions,
- unstable world-model predictions,
- constraint collapse.

Thus, reliability becomes a question of constraint resilience.

Operationally, this suggests:

1. Instrument transitions for constraint monitoring.
2. Distinguish between constraint satisfaction and constraint transformation.
3. Design feedback loops where agents adjust constraints explicitly.
4. Separate generative components from stabilizing constraint layers.

9.6 Agents as Dual Semantic Engines

Agents are both:

- meaning generators (producing state trajectories),
- constraint regulators (reshaping admissible futures).

CDD formalizes the second role.

Without structured constraint governance, agents risk semantic drift in open knowledge environments.

With CDD, constraint evolution becomes deliberate rather than emergent.

9.7 Toward Engineering Semantics

The proposed redefinition transforms semantics from a philosophical abstraction into an engineering discipline.

Semantics is no longer only about interpretation. It is about maintaining structural coherence across evolving state spaces.

In this view, Constraint-Driven Development is not merely a methodology. It is the practical embodiment of AI-era semantics.

10 Constraint Governance Architecture: Operationalizing AI-Era Semantics

If semantics is redefined as constraint-stabilized state transition under contextual deformation, then system architecture must explicitly incorporate constraint governance.

This leads to a structural reinterpretation of system design:

> Constraint-Driven Development (CDD) may be understood as a constraint governance architecture.

In classical software systems, constraints appear as validation rules, type systems, or logical invariants. They operate as correctness checks layered atop computation.

Under the proposed semantic framework, constraints are no longer secondary. They are primary semantic regulators.

1. From Validation to Governance Validation assumes:

- fixed rules,
- static interpretation,
- error detection after execution.

Governance assumes:

- evolving constraint structures,
- monitoring across transitions,
- dynamic adjustment mechanisms.

If context functions as a local coordinate system over state space, and constraint update acts as an operator, then governance becomes the architectural mechanism that manages this operator.

2. Constraint as First-Class Architectural Primitive In agent systems interacting with open environments, constraint must be treated as a first-class entity:

- explicitly represented,
- continuously monitored,
- capable of controlled transformation.

This includes:

- schema integrity,
- action feasibility,
- world-model consistency,
- safety invariants.

Without structured governance, constraint evolution becomes implicit and unstable. Semantic drift follows.

3. Agents as Dual Semantic Engines Agents perform two intertwined functions:

1. They generate trajectories in state space.
2. They participate in reshaping the constraint structure governing those trajectories.

Thus, agents are both meaning generators and constraint regulators.
CDD provides the architectural framework for the second function.

4. Engineering Implication Redefining semantics operationally implies:

- Transition monitoring replaces output inspection.
- Constraint resilience becomes a reliability metric.
- Constraint update mechanisms must be explicit.
- Context management must be architecturally visible.

CDD, in this interpretation, is not merely a development methodology. It is the engineering embodiment of AI-era semantics.

5. Scope of the Present Work The present paper introduces CDD only insofar as it clarifies the architectural consequences of redefining semantics.

A full treatment of Constraint-Driven Development— including formal models, implementation patterns, and industrial applications— is beyond the scope of this paper and will be addressed separately.

Here, CDD functions as a conceptual bridge: it translates a semantic redefinition into a governance-oriented system design principle.

Semantics in the age of AI is therefore not a layer atop computation. It is the structural governance of state transitions under evolving constraints.

11 Conclusion

The evolution of artificial intelligence compels a reexamination of the concept of *semantics*. In classical compiler theory, semantics was defined as a mapping from syntax to well-defined semantic objects. In formal semantic theory, meaning was grounded in truth conditions within structured models. Both frameworks presupposed closed domains and stable interpretation spaces.

Large Language Models disrupted this structure. In LLMs, meaning is not assigned to syntactic objects. Instead, context shapes probabilistic transitions in a high-dimensional state space. Semantics becomes constraint-conditioned transition rather than static denotation.

Agent architectures extend this shift further. Operating in open environments, agents must continuously update internal states, world models, and constraint structures. Here, context no longer functions merely as accumulated history. It becomes a dynamic local coordinate system over state space. Constraints cease to be fixed filters and instead evolve through operator-level updates.

We therefore propose the following redefinition:

> Semantics in the age of AI is the coupled evolution of state transitions and constraint structures under contextual deformation.

Meaning is not merely constraint satisfaction. It includes constraint transformation. It is not solely interpretation. It is governance of admissible trajectories in open systems.

This reinterpretation bridges philosophical semantics and system architecture. If semantics is constraint-stabilized state transition, then reliable AI systems must incorporate explicit constraint governance mechanisms.

Constraint-Driven Development, introduced here only at a conceptual level, emerges as the architectural expression of this semantic shift. A full treatment of its formalization and industrial application remains future work.

The central insight remains:

> In the age of AI, semantics is no longer a static mapping. It is the dynamic stabilization and evolution of possibility under constraint.

Such a perspective preserves the etymological core of semantics— meaning as structured significance— while rendering it operational for open, agentic, and context-dependent systems.