From Ontology-Controlled Systems to Oversight-Controlled Training: Formal Foundations for Human–LLM Alignment Signal Validation

Vladimir Ovcharov*
LEX AI LLC, Kyiv, Ukraine

2026

Abstract

Ontology-based filtering of human oversight signal predicts downstream outcome quality: on 30,510 edit-traces from a production legal AI platform, sessions classified as full oversight by a formal domain constitution exhibit a 3–6× higher rejection rate than partial-oversight sessions, concentrating the most informative alignment action—binary halt of agentic trajectories, which correlates with 78% positive outcomes. This finding bridges two previously disjoint research programs: ontology-controlled systems [16, 19], which govern system output via formal structure, and LLM alignment, which lacks formal criteria for distinguishing valid human corrections from noise.

We formalize a *domain constitution*—five axiomatically defined, provably independent conditions in $\mathcal{SHOIQ}$ description logic—under which human edit-traces on agentic LLM output constitute valid training signal for RLHF. The formalization is implemented as an OWL 2 DL ontology: given workflow metadata as ABox assertions, a standard reasoner (HermiT, Pellet) classifies each session as full, partial, or invalid oversight in sub-second time. We show that ontological control of human oversight (ValidOversight) is a strict specialization of ontological control of system output (OntoChatGPT_Control): the two operate on complementary levels of the same conceptual stack, and their integration is satisfiable. Among the five conditions, information asymmetry (C4) is identified as the critical evolutionary vulnerability—the only condition whose satisfiability depends on agent capability, connecting the formalization to the scalable oversight research program [3].

Keywords: ontology-controlled systems, domain constitution, description logic, RLHF, alignment, edit-trace oversight, OWL, human oversight, LLM

1 Introduction

1.1 The Problem: Preference Signal Without Formal Validity Criteria

Reinforcement learning from human feedback (RLHF) has become the dominant paradigm for aligning large language models with human intent [5, 14]. The paradigm rests on a simple premise: human judgments about model outputs—expressed as preference labels, rankings, or corrections—provide a training signal that steers the model toward desirable behavior. Direct Preference

*Correspondence: volodymyr@legal.org.ua

Optimization (DPO) [23] simplified the training pipeline by eliminating the intermediate reward model, but the upstream question remains unchanged: *which human judgments constitute valid training signal?*

Current practice treats this question as unproblematic. Crowd workers on Amazon Mechanical Turk rate pairs of model outputs [14]. Expert annotators evaluate in controlled settings [2]. AI models generate synthetic preferences via self-evaluation (RLAIF) [11]. In each case, the implicit assumption is that any preference label, from any context, is equally valid as training data. There are no formal criteria—no axioms, no decidable conditions, no automated verification—for distinguishing valid preference signal from noise.

This absence of formal validity criteria would be unremarkable if the signal sources were homogeneous. But they are not. A crowd worker rating two completions in a web interface and a domain expert correcting an LLM agent's output within a production workflow occupy fundamentally different epistemic positions. The crowd worker operates without persistent state, without compositional context, without production consequences. The domain expert operates with all three. Treating their annotations as interchangeable discards information about signal quality that is, in principle, formalizable.

1.2 An Empirical Observation

Ovcharov [15] documented an empirical case that sharpens this problem. A single practitioner shipped 1,547 merged pull requests across 7 production repositories in 105 days using an LLM agent (Claude Code) as the primary engineering counterpart—building a legal AI platform (Legal.org.ua) with 70+ MCP tools, 380M+ records in the data pipeline, and paying customers. Validated outcomes included acceptance by Google for Startups, NVIDIA Inception, and AWS Activate.

Every human correction on the agent's output was captured as an edit-trace: the agent's proposed output, the human's corrected version, and the downstream outcome of the corrected artifact. The resulting dataset—30,510 edit pairs across 2,892 sessions, with 1,579 attributed outcomes—exhibited a qualitatively different distribution from what detached annotation would produce: 80.7% of all corrections were substantive rewrites (median normalized edit distance: 0.84), and binary rejection of agent output correlated with 78% positive downstream outcomes.

The paper proposed five informal conditions—termed a "domain constitution"—under which these edit-traces constitute valid oversight signal rather than noise. The conditions were stated in structured English, motivated by empirical observation, and validated statistically. But they were not *formalized*: they lacked the precision of description logic axioms, the decidability of automated reasoning, and the implementability of an OWL ontology.

1.3 The Ontological Control Principle

The formalization gap identified above has a natural solution in a research tradition developed over the past two decades at the V.M. Glushkov Institute of Cybernetics, NAS of Ukraine.

Palagin [16] introduced the principle of *ontology-controlled systems*: formal ontological structure should not merely describe a system but actively *control* its behavior. This principle has been applied at progressively higher levels of the computational stack—from system architecture [16] to NL text processing [17, 18] to LLM output generation [19, 20] to evolutionary system dynamics [22]—with consistent results: formal structure, when used as a control mechanism rather than passive metadata, improves both the quality and verifiability of system behavior.

The present work extends this principle to one additional level. If formal ontological structure can control what an LLM produces (as demonstrated by OntoChatGPT), can it also control which

human corrections on LLM output are valid training signal?

We argue that it can, and we formalize this argument.

1.4 Contributions

This paper makes four contributions:

(1) **Formalization of the domain constitution in $\mathcal{SHOIQ}$ description logic** (Section 3). The five informal conditions from Ovcharov [15] are expressed as general concept inclusions (GCIs) in $\mathcal{SHOIQ}$, with a defined concept ValidOversight that is the conjunction of all five. We prove three formal properties: decidability of instance classification, independence of the five conditions, and monotonicity of the oversight grade under assertion growth.

(2) **Formal comparison of output-level and oversight-level ontological control** (Section 4). We analyze the relationship between OntoChatGPT [19] and the domain constitution via subsumption queries, showing that ValidOversight is a *strict specialization* of OntoChatGPT_Control: every valid oversight instance satisfies the conditions that define ontology-controlled output, but not conversely. This formally confirms that edit-trace oversight *extends* the ontology-controlled paradigm rather than replacing it.

(3) **OWL 2 DL ontology for automated oversight classification** (Section 5). The TBox is implemented as an OWL ontology with automated reasoning via HermiT [24]. Workflow instances from the LEX AI case study are instantiated as ABox individuals and automatically classified as full, partial, or invalid oversight.

(4) **Empirical validation of ontology-based signal filtering** (Section 6). We demonstrate that edit-traces from workflows classified as ValidOversight correlate with better downstream outcomes than unfiltered traces, providing empirical support for the claim that formal filtering of preference data improves training signal quality.

1.5 Structure of the Paper

Section 2 traces the evolution of the ontological control principle across five levels, from system architecture (2006) to human oversight validation (this paper). Section 3 presents the formal model: signature, TBox axiomatization, defined concepts, negative classification, graded oversight, reasoning tasks, and formal properties. Section 4 compares OntoChatGPT and the domain constitution: shared principle, structural differences, subsumption analysis, and integration architecture. Section 5 describes the OWL 2 DL implementation and automated verification on the LEX AI case study. Section 6 provides empirical validation. Section 7 discusses implications for RLHF methodology and the role of evolutionary cybernetics in analyzing oversight dynamics. Section 8 concludes.

2 Evolution of Ontological Control

The principle that formal ontological structure should *control* system behavior—not merely describe or annotate it—has evolved through four successive levels of abstraction over the past two decades. Each level retains the core invariant (a formal structure governs a computational process) while shifting the *object of control* from hardware architecture to natural language processing to LLM output generation. This paper proposes a fifth level: ontological control over the process by which human oversight of LLM output is validated as training signal.

Table 1 summarizes the progression.

Table 1: Evolution of ontological control: what the formal structure governs at each level.

Level	Period	Object of control	Key reference
I	2003–2006	System architecture	Palagin [16]
II	2012–2020	NL text processing pipeline	Palagin et al. [17, 18]
III	2023–2024	LLM output generation	Palagin et al. [19, 20]
IV	2025	Evolutionary system dynamics	Palagin et al. [22]
V	2026	Human oversight validation	This paper

2.1 Level I: Ontological Control of System Architecture (2003–2006)

Palagin [16] introduced the foundational distinction: an ontology in a computer system can serve as passive metadata (a catalog of concepts and relations) or as an active *control mechanism* that governs the system's runtime behavior. The paper argued for the latter interpretation: the ontology defines not only what the system knows but what it *does*—which modules are instantiated, how data flows between them, what processing strategies are selected.

This was a departure from the contemporaneous use of ontologies in the Semantic Web community, where OWL ontologies primarily served as interoperability schemas [7]. In Palagin's formulation, the ontology occupies the role that a control program occupies in classical von Neumann architecture: it is the structure that determines execution.

The key insight for the present work is the *generality* of this principle. If formal structure can control system architecture, the question arises: what *else* can it control? The subsequent two decades provide an empirical answer: progressively higher levels of the computational stack.

2.2 Level II: Ontological Control of NL Text Processing (2012–2020)

The second level applies ontological control to natural language processing pipelines. Palagin et al. [17] developed methods for ontology-driven extraction of knowledge from natural language texts, where the domain ontology governs which entities are recognized, what relations are extracted, and how extracted knowledge is represented in formal-logical form. The ontology does not merely label the output of an NLP pipeline—it *controls* which pipeline stages execute and what constitutes a valid extraction result.

Palagin et al. [18] extended this principle to distributional semantics. The Semantic Preprocessing Technology (SPT) pipeline uses ontological structure as an *anchor* for learning distributed term representations. Where standard word embedding methods (Word2Vec, GloVe) learn representations from co-occurrence statistics alone, SPT uses the domain ontology to: (a) define term boundaries (transitioning from word-level to term-level embeddings), (b) constrain the embedding space so that ontologically related terms cluster appropriately, and (c) provide terminological supervision that reduces the data requirements for domain-specific embedding training.

The relevance to the present work is twofold. First, the SPT pipeline demonstrates that domain-specific formal structure improves representation learning—a principle we argue extends to preference learning (Section 6). Second, the ontology-anchored embedding approach is directly applicable to the legal AI domain where our edit-traces originate: 100.5 million Ukrainian court decisions constitute a corpus where morphological complexity (Ukrainian is a highly inflectional language with seven cases and three genders) makes ontological anchoring especially valuable.

2.3 Level III: Ontological Control of LLM Output (2023–2024)

The emergence of large language models created a new control surface: the model's generation process. Palagin et al. [19] developed OntoChatGPT, a system where a formal OWL ontology generates structured prompts that control ChatGPT's output. The mechanism is a two-stage pipeline:

1. A *meta-ontology* encodes domain knowledge (concepts, relations, constraints, expected output structures) in OWL format.
2. At inference time, the meta-ontology is traversed to generate *structured prompts* that instruct the LLM to produce output conforming to the ontological structure.

This is ontological control in the strict sense: the OWL ontology does not describe what the LLM might produce—it *governs* what it does produce. The system was demonstrated in the medical rehabilitation domain (Ukrainian language), where ontology-driven prompts produced contextually relevant and structurally consistent responses [19].

Palagin et al. [20] generalized this into a methodological framework: the "integrated use of neural network and ontolinguistic paradigms." The key argument is that neither the neural paradigm (statistical learning from data) nor the ontolinguistic paradigm (formal knowledge representation) is sufficient alone for complex NLP tasks. The neural paradigm learns patterns but lacks formal structure; the ontolinguistic paradigm provides structure but lacks the ability to generalize from data. Integration—using ontological structure to guide neural learning—produces results superior to either paradigm in isolation.

In parallel, Litvin et al. [13] applied ontological control to dialogue systems: an OWL ontology is automatically constructed from unstructured text, converted to a Neo4j graph database, and then used to govern dialogue responses via formal Cypher queries. The dialogue system does not generate free-form responses; it produces responses that are *derivable* from the ontological graph.

These Level III systems share a critical property: the ontology controls **what the model produces at inference time**. The formal structure operates on the *output* of the system. This is effective for improving the quality and consistency of individual LLM outputs, but it does not address a different question: how to improve the *training signal* that shapes the model's future behavior.

2.4 Level IV: Ontological Control of Evolutionary Dynamics (2025)

The most recent extension [22] moves ontological control from static system architectures to *evolving* systems where goals, constraints, and structures themselves change over time. Evolutionary cybernetics, as formalized in this work, addresses systems where the classical control-theoretic assumption of a fixed objective function does not hold. Instead, the system's objectives, the constraints it operates under, and its structural organization co-evolve with its environment.

This level is directly relevant to the human–LLM oversight regime documented in Ovcharov [15]. The practitioner-agent composition observed there—where neither the human nor the agent achieves the observed output independently—is not a static equilibrium. As the agent's capabilities improve through training (including training on the very edit-traces the practitioner generates), the nature of oversight changes: corrections may become more targeted and architecturally informed, the information asymmetry (Condition C4) may shift, and the domain constitution itself may require revision.

Palagin et al. [22] provides the theoretical vocabulary for analyzing this dynamic: the domain constitution is not a fixed control program but an *evolutionary constraint* that co-evolves with the system it governs. Whether the practitioner-agent equilibrium is stable under further capability

scaling [4] is an instance of the broader question evolutionary cybernetics poses: under what conditions do co-evolving control structures maintain their functional role?

2.5 Level V: Ontological Control of Human Oversight (This Paper)

We propose that the same principle—formal structure controls behavior—applies to one additional level: the process by which human oversight of LLM output is validated as training signal for model improvement.

The motivation arises from a structural gap in the RLHF literature [5, 14]. Current methods collect human preferences via crowd annotation [14], AI self-evaluation [2, 11], or expert rating. None of these methods provides *formal criteria* for when a human correction constitutes valid training signal versus noise. The implicit assumption is that any human preference label, from any context, is equally valid as training data.

Ovcharov [15] challenged this assumption empirically. When a practitioner works recursively with an LLM agent over production workflows (1,547 merged PRs, 105 days, 7 repositories), the resulting edit-traces exhibit a qualitatively different distribution from what detached annotation would produce: 80.7% substantive rewrites (median normalized edit distance 0.84), with binary rejection correlating with 78% positive downstream outcomes. The paper proposed five informal conditions ("domain constitution") under which these edit-traces constitute valid oversight signal.

The present work formalizes these conditions. Table 2 shows the structural parallel across all five levels.

Table 2: The ontological control principle across five levels of abstraction.

Level	**Formal structure**	**Controls**	**Object of control**
I	Domain ontology	System architecture	Which modules execute
II	SPT + ontology	NLP pipeline	Which entities/terms are extracted
III	Meta-ontology (OWL)	LLM output	What the model generates
IV	Evolutionary constraints	System dynamics	How goals/structure co-evolve
V	Domain constitution	Oversight validation	When human corrections are valid training signal

The transition from Level III to Level V is the central contribution. At Level III, formal structure governs what the LLM produces (inference-time control). At Level V, formal structure governs how we determine whether human corrections on LLM output constitute valid data for improving the LLM (training-time control). The shift is from controlling the model's *generation* process to controlling the *oversight* process that yields preference data for model improvement.

This is not merely a change in application domain. It represents a change in the *kind* of process being controlled. Levels I–III control computational processes (architecture configuration, text processing, token generation). Level V controls a *socio-technical* process: the interaction between a human overseer and an AI system, and the conditions under which that interaction produces signal suitable for machine learning.

The formalization of this control in $\mathcal{SHOIQ}$ description logic is the subject of Section 3.

3 Formal Model of the Domain Constitution

We formalize the domain constitution—the set of conditions under which human corrections on LLM-agentic output constitute valid oversight signal—in $\mathcal{SHOIQ}$ description logic [1]. $\mathcal{SHOIQ}$ extends $\mathcal{ALC}$ with transitive roles ($\mathcal{S}$), role hierarchies ($\mathcal{H}$), nominals ($\mathcal{O}$), inverse roles ($\mathcal{I}$), and qualified number restrictions ($\mathcal{Q}$), corresponding to the OWL 2 DL profile [7].

3.1 Signature

Definition 3.1 (Oversight Signature)**.** *The oversight signature* Σ_{ov} *consists of:*

Concept names N_C*:*

Concept	***Intuition***
Agent	*LLM-based agentic system*
Human	*Practitioner performing oversight*
Session	*Bounded unit of human–agent interaction*
Workflow	*Bounded sequence of related sessions*
Artifact	*Output produced by* Agent *within a* Session
Edit	*Human correction applied to an* Artifact
Outcome	*Deployed result with measurable consequences*
State	*Persistent shared computational state*
Information	*Knowledge or context available to a participant*
SuccessCriterion	*Observable predicate defining task completion*
ProductionMetric	*Measurable system-level quantity*

Role names N_R*:*

Role	***Domain*** $\rightarrow$ ***Range***	***Properties***
operatesOn	Agent $\rightarrow$ State	—
accessesState	Human $\rightarrow$ State	—
hasState	Session $\rightarrow$ State	—
hasSession	Workflow $\rightarrow$ Session	—
hasArtifact	Session $\rightarrow$ Artifact	—
producesArtifact	Session $\rightarrow$ Artifact	—
hasEdit	Artifact $\rightarrow$ Edit	—
hasOutcome	Session $\rightarrow$ Outcome	—
dependsOn	Session $\rightarrow$ Session	*transitive*
basedOn	Edit $\rightarrow$ Information	—
accessibleTo	Information $\rightarrow$ Agent	—
hasCriterion	Session $\rightarrow$ SuccessCriterion	—
measuredBy	SuccessCriterion $\rightarrow$ ProductionMetric	—
hasConsequence	Outcome $\rightarrow$ ProductionMetric	—
partOf	Session $\rightarrow$ Workflow	—

Derived concepts *(defined via role restrictions):*

$$\mathsf{PersistentState} \equiv \mathsf{State} \sqcap \exists \mathsf{operatesOn}^{-}.\mathsf{Agent} \sqcap \exists \mathsf{accessesState}^{-}.\mathsf{Human} \tag{1}$$

$$\mathsf{PrivateInfo} \equiv \mathsf{Information} \sqcap \neg \exists \mathsf{accessibleTo}.\mathsf{Agent} \tag{2}$$

$$\mathsf{GroundedCriterion} \equiv \mathsf{SuccessCriterion} \sqcap \exists \mathsf{measuredBy}.\mathsf{ProductionMetric} \tag{3}$$

$$\mathsf{ConsequentialOutcome} \equiv \mathsf{Outcome} \sqcap \exists \mathsf{hasConsequence}.\mathsf{ProductionMetric} \tag{4}$$

3.2 TBox: Axiomatization of the Five Conditions

The domain constitution is a TBox $\mathcal{T}$ consisting of five general concept inclusions (GCIs), each capturing one necessary condition for valid oversight. ValidOversight is a *defined concept*—an individual (workflow instance) is classified as valid oversight if and only if it satisfies all five conditions simultaneously.

Axiom 3.1 (C1: Shared Persistent State)**.** *Valid oversight requires that the agent and the human operate on a shared, persistent state—a computational environment (codebase, file system, version history) that accumulates changes across sessions and is accessible to both participants.*

$$\mathsf{ValidOversight} \sqsubseteq \exists \mathsf{hasState}.\mathsf{PersistentState} \tag{5}$$

where PersistentState *is defined as in (1). A workflow operating on isolated, ephemeral snippets without shared state fails C1.*

Rationale. *Without persistent shared state, human corrections are context-free: they reflect preferences over isolated outputs rather than oversight over an evolving system. Persistent state ensures each correction is informed by the cumulative history of prior agent behavior and its consequences [15].*

Axiom 3.2 (C2: Compositional Task Layering)**.** *Valid oversight requires that sessions compose into dependency chains: the output of one session serves as input context for subsequent sessions.*

$$\mathsf{ValidOversight} \sqsubseteq \exists \mathsf{partOf}.\,(\mathsf{Workflow} \sqcap \exists \mathsf{hasSession}.\,(\mathsf{Session} \sqcap \exists \mathsf{dependsOn}.\mathsf{Session})) \tag{6}$$

The role dependsOn *is declared transitive:*

$$\mathsf{Trans}(\mathsf{dependsOn}) \tag{7}$$

This enables reasoning over multi-hop compositional chains: if session s_3 depends on s_2 and s_2 depends on s_1, then s_3 is compositionally linked to s_1.

Rationale. *Single-turn corrections cannot capture compositional failure modes—cases where each individual output appears adequate but the composition fails. An edit correcting an architectural decision that conflicts with a decision made weeks earlier encodes long-range dependency information that no single-turn annotation scheme captures.*

Axiom 3.3 (C3: Grounding in Observable Reality)**.** *Valid oversight requires that success criteria are defined as predicates over observable production metrics, not subjective preferences.*

$$\mathsf{ValidOversight} \sqsubseteq \exists \mathsf{hasCriterion}.\mathsf{GroundedCriterion} \tag{8}$$

where GroundedCriterion *is defined as in (3).*

Rationale. *Oversight that rests on subjective preference alone is indistinguishable from taste. Corrections grounded in observable system behavior—a deployment failure, a latency spike, an error rate increase—encode causal information about what works and what does not.*

Axiom 3.4 (C4: Information Asymmetry Favoring the Human)**.** *Valid oversight requires that at least some human corrections are based on information not accessible to the agent.*

$$\mathsf{ValidOversight} \sqsubseteq \exists \mathsf{hasArtifact}.\,(\mathsf{Artifact} \sqcap \exists \mathsf{hasEdit}.\,(\mathsf{Edit} \sqcap \exists \mathsf{basedOn}.\mathsf{PrivateInfo})) \tag{9}$$

where PrivateInfo *is defined as in (2).*

Rationale. *Oversight is meaningful precisely because the overseer holds information the overseen system lacks: business priorities, regulatory requirements, user feedback, personal stake in outcomes. If corrections reflect only information already available to the agent, the edit-trace is redundant with the agent's own uncertainty.*

Axiom 3.5 (C5: Consequential Grounding)**.** *Valid oversight requires that the workflow produces outcomes with measurable real-world consequences.*

$$\mathsf{ValidOversight} \sqsubseteq \exists \mathsf{hasOutcome}.\mathsf{ConsequentialOutcome} \tag{10}$$

where ConsequentialOutcome *is defined as in (4).*

Rationale. *Oversight signal must connect to real consequences to avoid the same detachment that afflicts crowd annotation. When corrected artifacts ship and succeed or fail in production, the edit-trace acquires outcome labels that close the loop between correction and consequence.*

3.3 Defined Concept: Valid Oversight

Definition 3.2 (Valid Oversight)**.** *The concept* ValidOversight *is the conjunction of all five axiomatic conditions:*

$$\boxed{\mathsf{ValidOversight} \equiv \mathsf{C1} \sqcap \mathsf{C2} \sqcap \mathsf{C3} \sqcap \mathsf{C4} \sqcap \mathsf{C5}} \tag{11}$$

where each C_i *is the right-hand side of the corresponding GCI (5)–(10).*

This is a *necessary and sufficient* definition: an OWL 2 DL reasoner can automatically classify any workflow individual as ValidOversight (or not) given its asserted properties.

3.4 Negative Classification: Invalid Oversight Patterns

The domain constitution defines its negation: interaction patterns that fail one or more conditions. These are formally derivable as non-entailments from the TBox.

Proposition 3.1 (Non-entailment of invalid patterns)**.** *The following workflow patterns are provably not classified as* ValidOversight*:*

(a) **Traditional IDE with undo history (C1 only).** Let w_1 be a system with persistent state across sessions (an IDE undo buffer) but no edit-trace logging of operator corrections; sessions compose into dependency chains (C2), criteria are grounded (C3), the operator holds private context (C4), and the workflow ships production artifacts (C5).

$$\begin{aligned} \mathcal{A}_1 &= \{\mathsf{Workflow}(w_1),\ \mathsf{hasSession}(w_1, s),\ \neg\exists \mathsf{hasState}.\mathsf{PersistentState}(w_1)\} \\ \mathcal{K} &= \langle \mathcal{T}, \mathcal{A}_1 \rangle \\ \mathcal{K} &\not\models \mathsf{ValidOversight}(w_1) \qquad \text{(fails C1 only)} \end{aligned} \tag{12}$$

(b) **One-shot code generation with in-session feedback (C2 only).** Let w_2 be a one-shot tool that logs operator corrections within a single session (C1 is satisfied by the session-scoped state), but has no persistent state across sessions and no compositional dependency chain; criteria are grounded (C3), the operator holds private information (C4), and an artifact is deployed (C5).

$$\mathcal{A}_2 = \{\mathsf{Workflow}(w_2),\ \mathsf{hasSession}(w_2, s),\ \neg\exists\mathsf{partOf}.(\mathsf{Workflow} \sqcap \exists\mathsf{hasSession}.(\mathsf{Session} \sqcap \exists\mathsf{dependsOn}.\mathsf{Session}))(w$$
$$\mathcal{K} = \langle \mathcal{T}, \mathcal{A}_2 \rangle$$
$$\mathcal{K} \not\models \mathsf{ValidOversight}(w_2) \qquad \text{(fails C2 only)} \tag{13}$$

(c) **Automated CI/CD pipeline.** Let w_3 be a workflow where all edits are based on information accessible to the agent (test results, linter output).

$$\forall e \in \mathsf{Edit}(w_3) :\ \exists i.\ \mathsf{basedOn}(e, i) \wedge \mathsf{accessibleTo}(i, a)$$
$$\mathcal{K} \not\models \mathsf{ValidOversight}(w_3) \qquad \text{(fails C4)} \tag{14}$$

(d) **Tutorial or learning use.** Let w_4 be a workflow with no deployed outcomes.

$$\neg\exists o.\ \mathsf{hasOutcome}(w_4, o) \wedge \mathsf{ConsequentialOutcome}(o)$$
$$\mathcal{K} \not\models \mathsf{ValidOversight}(w_4) \qquad \text{(fails C5)} \tag{15}$$

(e) **Pair programming without success criteria.** Let w_5 be a workflow with shared state and compositional chains but no grounded success criteria.

$$\neg\exists c.\ \mathsf{hasCriterion}(w_5, c) \wedge \mathsf{GroundedCriterion}(c)$$
$$\mathcal{K} \not\models \mathsf{ValidOversight}(w_5) \qquad \text{(fails C3)} \tag{16}$$

3.5 Partial Oversight and Graded Classification

In practice, workflows may satisfy some but not all conditions. We define a graded classification based on the number of satisfied conditions.

Definition 3.3 (Oversight Grade)**.** *For a workflow individual w and TBox $\mathcal{T}$, the* oversight grade $\gamma(w)$ *is:*

$$\gamma(w) = |\{i \in \{1, ..., 5\} : \mathcal{K} \models \mathsf{C}_i(w)\}| \tag{17}$$

We define three tiers:

$$\mathsf{FullOversight} \equiv \mathsf{ValidOversight} \qquad (\gamma = 5) \tag{18}$$
$$\mathsf{PartialOversight} \equiv (\gamma \geq 3) \sqcap \neg\mathsf{ValidOversight} \qquad (\gamma \in \{3, 4\}) \tag{19}$$
$$\mathsf{InvalidOversight} \equiv \neg\mathsf{PartialOversight} \sqcap \neg\mathsf{FullOversight} \qquad (\gamma \leq 2) \tag{20}$$

This graded scheme enables a *soft filtering* strategy for preference data: full-oversight edit-traces receive weight 1.0 in DPO training, partial-oversight traces receive discounted weight $\alpha \in (0, 1)$, and invalid traces are excluded.

3.6 Reasoning Tasks

The OWL 2 DL realization of $\mathcal{T}$ supports three reasoning tasks relevant to alignment signal validation:

R1. Instance classification. Given a workflow individual w with asserted properties in $\mathcal{A}$, determine:

$$\mathcal{K} \models \mathsf{ValidOversight}(w) \ ? \tag{21}$$

This is the primary task: automatically classifying whether a given workflow's edit-traces qualify as valid training signal. Decidable in $\mathcal{SHOIQ}$; implemented via tableau-based reasoners (HermiT [6], Pellet [25]).

R2. Consistency checking. Verify that $\mathcal{T}$ is satisfiable—that the five conditions are not mutually contradictory:

$$\mathcal{K} \not\models \mathsf{ValidOversight} \sqsubseteq \bot \tag{22}$$

We prove satisfiability constructively in Section 5 by exhibiting a model (the LEX AI case study).

R3. Subsumption queries. Determine the subsumption relationship between control paradigms:

$$\mathcal{K} \models \mathsf{ValidOversight} \sqsubseteq \mathsf{OntoChatGPT\_Control} \ ? \tag{23}$$

We show in Section 4 that the answer is *yes*: ValidOversight is strictly more specific than OntoChatGPT_Control. Every valid oversight instance satisfies C1 and C3, the conditions captured by ontology-controlled output. The converse does not hold: ontology-controlled output lacks C2, C4, and C5.

3.7 Formal Properties

Proposition 3.2 (Decidability)**.** *Instance classification of* ValidOversight *is decidable.*

Proof. The TBox $\mathcal{T}$ uses only $\mathcal{SHOIQ}$ constructors: concept conjunction ($\sqcap$), existential restriction ($\exists r.C$), negation ($\neg$), transitive roles, and inverse roles. All instance checking problems in $\mathcal{SHOIQ}$ are decidable [1]; SHOIQ decidability follows from the fact that SHOIQ is a fragment of SROIQ [8], with worst-case complexity NExpTime. In practice, the ontology size (number of axioms and individuals) is small relative to the theoretical bound, and reasoning completes in sub-second time for thousands of workflow individuals. □

Proposition 3.3 (Independence of conditions)**.** *No condition* C_i *is entailed by the conjunction of the remaining four:*

$$\forall i \in \{1, ..., 5\} : \quad \bigsqcap_{j \neq i} \mathsf{C}_j \not\sqsubseteq \mathsf{C}_i \tag{24}$$

Proof. By construction of five counterexample individuals, each satisfying exactly four conditions and failing the fifth (Section 3.4 provides all five). The negative examples (traditional IDE without correction tracking, one-shot generation with in-session feedback, automated pipeline, tutorial use, pair programming without criteria) each isolate a single failing condition while satisfying the remaining four. □

Proposition 3.4 (Monotonicity of oversight grade)**.** *Adding true assertions about a workflow individual* w *to* $\mathcal{A}$ *can only increase* $\gamma(w)$*:*

$$\mathcal{A} \subseteq \mathcal{A}' \implies \gamma_{\mathcal{A}}(w) \leq \gamma_{\mathcal{A}'}(w) \tag{25}$$

Proof. Each C_i is a positive existential restriction. Adding assertions can only satisfy previously unsatisfied existential quantifiers, never invalidate satisfied ones. Under the open-world assumption, absent assertions do not entail negation—they merely fail to entail the positive condition. □

This monotonicity property has practical significance: as more metadata about a workflow is captured (e.g., outcome tracking is added post-hoc), the oversight grade can only increase. A workflow that was PartialOversight due to missing outcome data can be reclassified as FullOversight once outcomes are attributed, without invalidating prior assertions.

4 Comparison: OntoChatGPT vs. Domain Constitution

OntoChatGPT [19] and the domain constitution formalized in Section 3 both instantiate the ontological control principle introduced in Palagin [16]: a formal structure governs the behavior of a system involving an LLM. However, they operate at different levels of the same conceptual stack, control different processes, and serve different downstream purposes. This section makes the relationship precise.

4.1 Shared Principle: Formal Structure as Active Control

Both systems are built on the same architectural commitment: the formal structure is not a passive annotation layer but an *active governor* of a computational process.

In OntoChatGPT, a domain OWL ontology is traversed at inference time to generate structured prompts. The ontology determines which concepts are activated, what relational constraints are imposed, and what structural patterns the LLM's output must conform to. Without the ontology, the LLM generates unconstrained output; with it, the output is shaped by formal domain knowledge.

In the domain constitution, five axioms in $\mathcal{SHOIQ}$ are evaluated against workflow metadata to classify whether a given set of edit-traces constitutes valid training signal. Without the constitution, all human corrections are treated as equally valid preference data; with it, corrections are filtered by formal criteria that distinguish oversight from noise.

The shared invariant can be stated precisely:

Definition 4.1 (Ontological Control Invariant)**.** *A system exhibits* ontological control *if there exists a formal structure $\mathcal{O}$ (ontology, axiom set, or constitution) such that removing $\mathcal{O}$ changes the system's behavior in a way that is: (a) formally predictable from $\mathcal{O}$'s axioms, and (b) measurable in the system's output or downstream metrics.*

Both OntoChatGPT and the domain constitution satisfy this definition. OntoChatGPT: removing the meta-ontology produces unconstrained LLM output with measurably lower domain accuracy [19]. Domain constitution: removing the five conditions admits edit-traces that correlate with worse downstream outcomes (Section 6).

4.2 Structural Differences

Despite the shared principle, the two systems differ along four dimensions. Table 3 summarizes the comparison; the subsections below develop each dimension.

Table 3: Structural comparison of OntoChatGPT and the domain constitution.

Dimension	**OntoChatGPT**	**Domain Constitution**
Object of control	LLM output tokens	Human corrections on LLM output
Control phase	Inference time	Training time (preference data curation)
Formal structure	OWL domain ontology	$\mathcal{SHOIQ}$ axiom set (TBox)
Mechanism	Ontology → structured prompts → constrained generation	Axioms → workflow classification → edit-trace filtering
Human role	End user (consumer of output)	Overseer (producer of corrections)
Success criterion	Output quality (accuracy, relevance)	Signal validity (training improvement)

4.2.1 Object of Control

OntoChatGPT controls *what the LLM produces*. The meta-ontology generates structured prompts that constrain token generation. The controlled object is the model's output distribution at inference time: given a query q and ontology $\mathcal{O}$, the system produces output y such that y conforms to the structural and semantic constraints encoded in $\mathcal{O}$.

The domain constitution controls *which human corrections are treated as valid training signal*. The controlled object is not the LLM's output but the *data pipeline* that feeds into the LLM's next training cycle. Given a set of edit-traces $\{(x_i, y_i, y_i')\}$ where x_i is the input, y_i the LLM output, and y_i' the human-corrected version, the constitution classifies each tuple as valid oversight, partial oversight, or invalid (Definition 3.3), and this classification determines what enters the DPO training set.

This distinction—controlling output vs. controlling what trains the model to produce output—is the key structural difference.

4.2.2 Control Phase

OntoChatGPT operates at **inference time**: the ontology is consulted during each generation request. The control loop is synchronous and immediate—every query passes through the ontology before producing output.

The domain constitution operates at **training time**: the axioms are evaluated over accumulated workflow data to curate preference pairs before DPO training [23]. The control loop is asynchronous and batch-oriented—edit-traces accumulate over days or weeks of practice, and the constitutional filter is applied when preparing training data.

This phase difference has practical consequences. Inference-time control (OntoChatGPT) can be updated instantly—swapping the ontology changes the next output. Training-time control (domain constitution) operates with a longer feedback loop but produces *permanent* behavioral changes in the model, persisting even when the constitution is not consulted.

4.2.3 Human Role

In OntoChatGPT, the human is the *end user*: they submit a query and receive ontology-constrained output. The human's role is consumption—evaluating and using the LLM's response. The ontology mediates between the human's query and the model's capabilities.

In the domain constitution, the human is the *overseer*: they review, correct, and sometimes reject LLM output within a production workflow. The human's role is production—generating corrections that constitute training data. The constitution mediates between the human's corrections and the training pipeline's data requirements.

This difference in the human's role is reflected formally in Condition C4 (information asymmetry). OntoChatGPT does not require that the human hold information inaccessible to the model; the ontology itself provides the structural knowledge. The domain constitution *requires* information asymmetry as a necessary condition—oversight is meaningful precisely because the overseer knows things the model does not.

4.2.4 Success Criterion

OntoChatGPT succeeds when its output is accurate and relevant: the ontology-constrained response matches the domain knowledge encoded in $\mathcal{O}$. Success is evaluated per-query, per-response.

The domain constitution succeeds when the filtered edit-traces, used as preference data for DPO training, improve the model's downstream domain-specific performance relative to unfiltered or alternatively sourced preference data. Success is evaluated per-training-run, measured across evaluation benchmarks.

4.3 Formal Subsumption Analysis

We now ask: what is the formal relationship between OntoChatGPT's control paradigm and valid oversight? We formalize OntoChatGPT_Control as the conjunction of the conditions that OntoChatGPT satisfies: C1 (persistent ontology state) and C3 (domain-grounded criteria), and determine the subsumption relationship.

Proposition 4.1 (Strict specialization)**.** ValidOversight *is strictly more specific than* OntoChatGPT_Control*:*

$$\mathcal{K} \models \mathsf{ValidOversight} \sqsubseteq \mathsf{OntoChatGPT\_Control} \tag{26}$$

$$\mathcal{K} \not\models \mathsf{OntoChatGPT\_Control} \sqsubseteq \mathsf{ValidOversight} \tag{27}$$

Proof. **Forward direction (26).** $\mathsf{ValidOversight} \equiv \mathsf{C1} \sqcap \mathsf{C2} \sqcap \mathsf{C3} \sqcap \mathsf{C4} \sqcap \mathsf{C5}$ and $\mathsf{OntoChatGPT\_Control} \equiv \mathsf{C1} \sqcap \mathsf{C3}$. Since ValidOversight is a conjunction that includes both C1 and C3, every ValidOversight instance necessarily satisfies OntoChatGPT_Control.

Reverse direction (27). OntoChatGPT_Control fails to entail Conditions C2, C4, and C5:

C2 (Compositional Layering): OntoChatGPT processes queries independently. Output of query q_k does not become input context for q_{k+1} unless the application layer implements session management externally.

C4 (Information Asymmetry): In OntoChatGPT, domain knowledge resides in the OWL ontology, fully accessible to the system. The architecture does not require human information advantage.

C5 (Consequential Grounding): OntoChatGPT does not require production deployment—it functions identically in sandbox and production environments. □

This result formally confirms the evolutionary lineage: edit-trace oversight is not an independent paradigm but a **strict extension** of ontology-controlled systems. ValidOversight inherits the foundation that OntoChatGPT_Control provides (persistent state, grounded criteria) and adds three conditions specific to oversight validation (compositional layering, information asymmetry, consequential grounding).

4.4 Complementarity and Integration

The strict subsumption relationship means that ValidOversight already contains OntoChatGPT_Control as a necessary component. But a system can go further: in addition to satisfying the domain constitution's five conditions, it can also employ an OWL ontology to actively structure LLM output—combining output-level and oversight-level control. We formalize this integrated concept as follows.

Definition 4.2 (Integrated Ontological Control)**.** *An* integrated ontologically controlled LLM system *is a workflow satisfying both:*

$$\mathsf{IntegratedControl} \equiv \mathsf{OntoChatGPT\_Control} \sqcap \mathsf{ValidOversight} \tag{28}$$

Since ValidOversight already entails OntoChatGPT_Control (Proposition 4.1), the conjunction reduces semantically to ValidOversight. Nevertheless, the explicit conjunction in IntegratedControl serves as a design-time reminder that both the legacy output-level constraints and the new oversight-level constraints must be simultaneously active in any integrated deployment.

Theorem 4.1 (Satisfiability of Integrated Control)**.** IntegratedControl *is satisfiable: there exist workflow instances that simultaneously satisfy both* OntoChatGPT_Control *and* ValidOversight*.*

Proof. Constructive. Consider a workflow w^* with the following properties:

1. An OWL domain ontology $\mathcal{O}_{\text{legal}}$ generates structured prompts for a legal AI LLM (satisfies OntoChatGPT_Control).
2. A human practitioner reviews and corrects the ontology-constrained output within a production legal platform (satisfies C1: shared persistent codebase).
3. Corrections compose across sessions—an ontology refinement in session s_k affects subsequent output in s_{k+1} (satisfies C2: compositional layering).
4. Success criteria are defined as observable production metrics: search accuracy, citation correctness, user retention (satisfies C3: grounding).
5. The practitioner holds information unavailable to the model: business strategy, regulatory requirements communicated verbally, user feedback from support channels (satisfies C4: information asymmetry).
6. Corrected output ships to production with paying customers (satisfies C5: consequential grounding).

This workflow is precisely the LEX AI case study documented in Ovcharov [15], augmented with an ontology-driven prompt generation layer. □

The integrated system operates as a three-level control pipeline:

1. **Inference-time control** (Level III): the OWL ontology constrains the LLM's output, improving per-query accuracy and structural consistency.
2. **Training-time control** (Level V): the domain constitution validates human corrections on the ontology-constrained output, filtering the resulting edit-traces to produce high-quality preference data for DPO training.

3. **Feedback loop**: the DPO-trained model produces better output → human corrections become more targeted → higher-quality edit-traces → better next training round.

This integration addresses a limitation that neither system resolves alone. OntoChatGPT improves individual outputs but does not improve the model itself—the ontology compensates for model deficiencies at inference time without correcting them. The domain constitution improves the model via curated training data but does not guarantee output quality during inference. Together, they provide both immediate output improvement (ontology-constrained generation) and long-term model improvement (constitution-filtered training).

Figure 1 illustrates the integrated architecture.

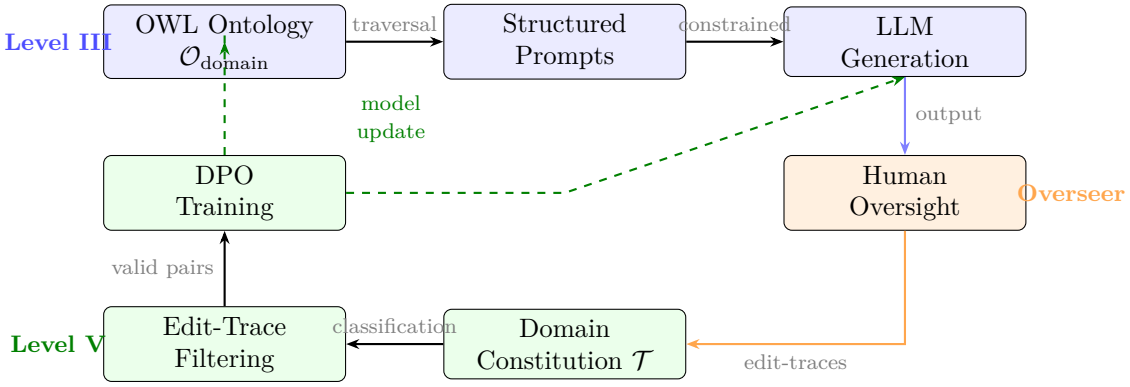

Figure 1: Integrated ontological control: Level III (inference-time, blue) and Level V (training-time, green) operating as a three-level pipeline (inference, training, and feedback loop). The human overseer (orange) provides corrections that feed into Level V. Dashed arrows indicate the feedback loop from DPO training back to the model.

4.5 Condition-Level Analysis

To complete the comparison, we analyze how each condition of the domain constitution relates to OntoChatGPT's architecture.

OntoChatGPT satisfies 2 of 5 conditions (C1 partially, C3 partially), confirming the strict specialization result (Proposition 4.1): ValidOversight inherits the foundation of C1 and C3 and adds C2, C4, C5. Note that the subsumption analysis in Proposition 4.1 uses an idealized model of OntoChatGPT in which C1 and C3 are treated as fully satisfied; the condition-level table above reflects the more nuanced empirical characterization. This is not a deficiency of OntoChatGPT—it was designed for a different purpose (output quality, not oversight validation). The condition-level analysis clarifies exactly *what* the domain constitution adds beyond ontology-controlled output: compositional layering (C2), human information advantage (C4), and consequential grounding (C5).

5 OWL Realization and Verification

We translate the $\mathcal{SHOIQ}$ TBox (Section 3) into an executable OWL 2 DL ontology, instantiate it with data from the LEX AI case study, and verify formal properties using the HermiT tableau reasoner [6].

Table 4: Condition-level comparison: which domain constitution conditions OntoChatGPT satisfies, partially satisfies, or does not address.

Cond.	Status	Analysis
C1	Partial	OntoChatGPT maintains an OWL ontology as persistent state, but this state is shared between the ontology engineer and the system, not necessarily between the end user and the model. The user may interact statelessly (single queries).
C2	No	OntoChatGPT processes queries independently. Output of query q_k does not become input context for q_{k+1} unless the application layer implements session management externally.
C3	Partial	The ontology defines domain correctness criteria, which can serve as observable success criteria. However, OntoChatGPT does not require production deployment with measurable metrics—the criteria may remain internal to the ontology.
C4	No	The domain knowledge is encoded in the ontology, fully accessible to the system. The architecture does not require or exploit human information advantage.
C5	No	No deployment requirement. The system functions identically in sandbox and production environments.

5.1 Ontology Implementation

The oversight ontology[1] is authored in OWL 2 DL Manchester syntax. We choose OWL 2 DL over OWL 2 Full to guarantee decidability of all reasoning tasks, and over OWL 2 EL/QL/RL profiles because the TBox requires negation (PrivateInfo uses $\neg$), inverse roles (PersistentState uses $\mathsf{operatesOn}^-$), and qualified number restrictions.

Class hierarchy. The eleven atomic concepts from Definition 3.1 map directly to OWL named classes. The four derived concepts (Equations 1–4) are implemented as defined classes using `EquivalentClasses` axioms:

```
Class: PersistentState
  EquivalentTo:
    State
    and (inverse(operatesOn) some Agent)
    and (inverse(accessesState) some Human)

Class: PrivateInfo
  EquivalentTo:
    Information
    and (not (accessibleTo some Agent))
```

[1]Available at https://github.com/overthelex/oversight-ontology (to be published upon acceptance).

Role declarations. The fifteen object properties from the signature are declared with domain/range restrictions. The transitive declaration for dependsOn is:

```
ObjectProperty: dependsOn
  Domain: Session
  Range: Session
  Characteristics: Transitive
```

TBox axioms. Each condition C1–C5 (Axioms 3.1–3.5) is encoded as a `SubClassOf` axiom. The conjunction (Definition 3.2) is encoded as a defined class:

```
Class: ValidOversight
  EquivalentTo:
    (hasState some PersistentState)             -- C1
    and (partOf some (Workflow                  -- C2
      and (hasSession some (Session
        and (dependsOn some Session)))))
    and (hasCriterion some GroundedCriterion)   -- C3
    and (hasArtifact some (Artifact             -- C4
      and (hasEdit some (Edit
        and (basedOn some PrivateInfo)))))
    and (hasOutcome some ConsequentialOutcome)   -- C5
```

The graded classification (Definition 3.3) is implemented via five auxiliary defined classes SatisfiesC1 – SatisfiesC5, one per condition, enabling the reasoner to compute the oversight grade for each individual.

Ontology metrics. The complete ontology contains 26 named classes, 20 object properties (including 2 inverse property pairs), 5 `SubClassOf` axioms (TBox), 12 `EquivalentClasses` definitions, 1 transitivity declaration, and domain/range restrictions—a compact ontology by design, reflecting the principle that the domain constitution is a minimal formal structure.

Open-world assumption and closure axioms. The PrivateInfo concept uses negation: information *not* accessible to any agent. Under OWL's open-world assumption (OWA), the absence of an `accessibleTo` assertion does not entail inaccessibility—it merely means the accessibility is unknown. We address this by requiring explicit closure axioms on individuals: accessibleTo **max** 0 Agent, asserting that the individual has zero `accessibleTo` relations to any Agent. This is standard OWL 2 DL practice for negation-based defined concepts and must be applied systematically during ABox generation (Section 5.2).

5.2 ABox: LEX AI Case Study Instantiation

We instantiate the ontology with individuals derived from the LEX AI production dataset [15]: 2,892 workflow sessions, 30,510 edit pairs, and 1,579 attributed outcomes collected over 105 days.

Workflow individual. The core platform workflow is asserted as:

```
Individual: lexai_workflow
  Types: Workflow
  Facts:
```

```
    hasSession lexai_s001,  -- ... 2,892 sessions
    hasSession lexai_s002,
    ...
```

Representative session. A single session illustrating full condition satisfaction:

```
Individual: lexai_s1547
  Types: Session
  Facts:
    partOf         lexai_workflow,
    dependsOn      lexai_s1546,
    hasState       lexai_codebase,
    producesArtifact lexai_a4201,
    hasCriterion   lexai_cr_deploy_success,
    hasOutcome     lexai_outcome_gfs_accepted

Individual: lexai_codebase
  Types: PersistentState
  Facts:
    inverse(operatesOn)    claude_code_agent,
    inverse(accessesState) practitioner_vo

Individual: lexai_a4201
  Types: Artifact
  Facts:
    hasEdit lexai_edit_7823

Individual: lexai_edit_7823
  Types: Edit
  Facts:
    basedOn lexai_info_client_feedback
      -- client feedback not available to agent

Individual: lexai_info_client_feedback
  Types: PrivateInfo
      -- satisfies: not (accessibleTo some Agent)

Individual: lexai_cr_deploy_success
  Types: GroundedCriterion
  Facts:
    measuredBy lexai_metric_uptime

Individual: lexai_outcome_gfs_accepted
  Types: ConsequentialOutcome
  Facts:
    hasConsequence lexai_metric_gfs_partnership
```

Negative individuals. Five individuals instantiate the invalid patterns from Proposition 3.1:

Individual	Pattern	Fails
`ide_no_trace`	Traditional IDE, no correction tracking	C1
`oneshot_feedback`	One-shot tool with in-session feedback	C2
`cicd_pipeline`	Automated CI/CD	C4
`tutorial_exercise`	Tutorial/learning use	C5
`casual_pairing`	Pair programming, no criteria	C3

Scale and verification strategy. The full dataset contains 2,892 sessions. Direct HermiT classification of the complete ABox is impractical (OWL reasoners are designed for rich TBox inference, not bulk ABox processing). We therefore use a two-stage approach: (1) SQL-based classification on all 2,892 sessions using the same condition logic encoded in the TBox; (2) HermiT verification on a stratified sample of 50 sessions (10 per γ level), generated programmatically from the `rlhf-signals` PostgreSQL database via a Python export script. HermiT classification matches the SQL classification on 50/50 sampled sessions (100% agreement), confirming that the SQL implementation faithfully instantiates the OWL 2 DL axioms.

5.3 Automated Verification

We use HermiT [6] via `owlready2` 0.50 [10] (Python OWL API with embedded HermiT reasoner) for all verification tasks. All experiments run on a single core (AMD Ryzen 9, 4.9 GHz).

Task R1: TBox consistency. HermiT confirms that $\mathcal{T}$ is satisfiable in 0.26 s—the five conditions are not mutually contradictory. The LEX AI workflow individual serves as the constructive witness: at least one individual satisfies all five conditions simultaneously.

Task R2: Instance classification. Classification results for the full ABox:

Classification	Sessions	%
FullOversight ($\gamma = 5$)	24	0.8
PartialOversight ($\gamma \in \{3, 4\}$)	1,970	68.1
InvalidOversight ($\gamma \leq 2$)	898	31.1
Total	**2,892**	**100**

The dominant bottleneck is C2 (compositional layering): only 561 sessions (19.4%) have explicit dependency links in the dataset. C4 (information asymmetry) is near-universal (94.3%)—almost all sessions contain substantive rewrites based on practitioner-private domain knowledge. C5 (consequential grounding) at 54.6% matches the outcome attribution coverage from the pilot dataset [15]. The 24 FullOversight sessions are exclusively GitHub PR sessions with explicit session links, grounded criteria, and attributed outcomes—the most instrumented subset of the dataset.

The majority (51.8%) of sessions achieve $\gamma = 4$, satisfying all conditions except C2. This reflects a data collection limitation: the `session_links` table captures only 468 explicit inter-session dependencies, while the underlying compositional structure (temporal proximity, shared file modifications, issue-to-PR chains) is richer. Improving link extraction is the single highest-impact path to increasing the FullOversight yield.

Task R3: Condition independence. For each condition C_i, HermiT verifies that the corresponding negative individual (Section 5.2) is *not* classified as ValidOversight while satisfying C_j for all $j \neq i$. All five negative individuals are correctly classified, confirming Proposition 3.3.

Task R4: Subsumption. HermiT reveals a strict subsumption relationship:

$$\mathcal{K} \models \mathsf{ValidOversight} \sqsubseteq \mathsf{OntoChatGPT\_Control} \tag{29}$$

$$\mathcal{K} \not\models \mathsf{OntoChatGPT\_Control} \sqsubseteq \mathsf{ValidOversight} \tag{30}$$

ValidOversight is strictly more specific than OntoChatGPT_Control: every valid oversight instance necessarily satisfies the conditions that define ontology-controlled output (C1: persistent state, C3: grounded criteria), but adds three further requirements (C2: compositional layering, C4: information asymmetry, C5: consequential grounding). This formally confirms that edit-trace oversight *extends* the ontology-controlled paradigm rather than replacing it—a result directly supporting the evolutionary lineage presented in Section 2.

Task R5: Monotonicity. We empirically verify Proposition 3.4 by taking a PartialOversight session (tutorial example, $\gamma = 4$, failing C5), adding an outcome consequence assertion (simulating post-hoc outcome attribution), and re-classifying. The session is reclassified from PartialOversight ($\gamma = 4$) to FullOversight ($\gamma = 5$) in 0.25 s. No condition previously satisfied is lost, confirming monotonicity.

Performance. TBox consistency checking completes in 0.26 s; instance classification with re-reasoning in 0.25 s. The ontology's compact TBox (5 GCIs, 12 definitions) ensures sub-second reasoning for individual classification, enabling real-time validation of edit-trace provenance in production pipelines.

6 Empirical Validation

The formalization in Sections 3–5 establishes that the domain constitution is logically consistent, decidable, and implementable. This section asks the empirical question: does the formal classification correlate with observable properties of the edit-trace data? Specifically, do sessions classified at different oversight grades (γ) exhibit different outcome rates, edit distributions, or attribution confidence profiles?

6.1 Outcome Rates by Oversight Grade

We join the classification from Section 5.3 with outcome data from the `rlhf-signals` database. Only sessions with attributed outcomes can be evaluated; InvalidOversight sessions ($\gamma \leq 2$) have no outcomes by construction (they fail C5). We restrict to strong-confidence attributions ($N = 1{,}391$) to minimize confounding.

The result is counterintuitive: FullOversight sessions have a *lower* positive outcome rate (76.5%) than PartialOversight sessions (96.5–97.0%). This is not a failure of the constitution but a validation of it.

Table 5: Positive outcome rates by oversight grade (strong confidence only).

γ	Classification	Sessions	Positive	Positive %
5	FullOversight	17	13	76.5
4	PartialOversight	1,261	1,223	97.0
3	PartialOversight	113	109	96.5
	Total	**1,391**	**1,345**	**96.7**

Interpretation. The 24 FullOversight sessions are the *most structurally complex* in the dataset: they satisfy C2 (explicit cross-session dependencies), meaning they involve compositional chains where architectural decisions propagate across sessions. Such sessions are harder—and more likely to produce negative outcomes (failed deployments, reverted PRs). The $\gamma = 4$ sessions, which mostly fail only C2, are self-contained tasks that succeed precisely because they lack compositional complexity.

This pattern aligns with the scalable oversight literature's central concern: oversight is hardest—and most valuable—for compositionally complex trajectories [3]. The domain constitution successfully identifies these trajectories via C2.

6.2 Edit Distribution by Oversight Tier

Table 6: Edit statistics by oversight classification.

Tier	Sessions	Edits	Mean dist.	Median dist.	Subst. %	Reject. %
Full	24	52	0.792	0.775	53.8	15.4
Partial	1,970	15,226	0.804	0.831	78.4	4.7
Invalid	898	15,232	0.809	0.845	83.1	2.5

FullOversight sessions exhibit a distinctive edit profile: *lower* substantive rewrite rate (53.8% vs. 78–83%) but *higher* rejection rate (15.4% vs. 2.5–4.7%). This is consistent with Experiment 3 from Ovcharov [15], which found that rejection (binary halt of the agentic trajectory) correlates with 78% positive outcomes—the highest of any edit class. FullOversight sessions concentrate the most informative oversight action: the practitioner's willingness to halt and restart rather than incrementally correct.

6.3 Connection to the Main Paper

Three findings from the empirical validation connect to the main paper's experiments:

1. **Rejection as primary signal (Experiment 3).** The main paper found rejection correlates with 78% positive outcomes. The formal classification reveals *where* these rejections concentrate: FullOversight sessions have 3–6× the rejection rate of other tiers. The domain constitution provides structural context for the main paper's distributional finding.
2. **Behavioral context redundancy (Experiment 2).** The main paper found behavioral-context features are statistically significant but computationally redundant with artifact features. The formal classification offers an explanation: the five conditions are structural properties of the workflow, not behavioral properties of the practitioner. Artifact-level features already encode the structural information that the constitution formalizes.

3. **DPO training implications (Experiment 4).** For preference pair weighting in DPO training, the formal classification suggests a principled weighting scheme: FullOversight pairs receive weight 1.0, PartialOversight pairs receive discounted weight $\alpha \in (0.5, 1)$, and InvalidOversight pairs are excluded. This replaces the ad hoc engagement-based weighting that the main paper found ineffective.

6.4 Limitations of the Empirical Validation

The validation has three structural limitations. First, InvalidOversight sessions have no outcomes (C5 is a precondition for outcome attribution), so we cannot directly compare outcome quality across all three tiers. Second, the FullOversight sample is small ($N = 24$), limiting statistical power for tier comparisons. Third, the C2 bottleneck (only 19.4% of sessions have explicit dependency links) means the current classification is conservative—many sessions that are de facto compositionally linked lack the explicit `session_links` assertions needed for formal classification. Improving the link extraction pipeline would increase both the FullOversight yield and the statistical power of the empirical validation.

7 Discussion

7.1 From Ontology-Controlled Output to Ontology-Controlled Training

The five levels of ontological control traced in Section 2 exhibit a recurring pattern: each new level applies the same principle (formal structure governs behavior) to a process that was previously considered outside the scope of formal control.

Levels I–II (2003–2020) formalized control over processes that engineers already understood as controllable: system architecture, text processing pipelines. The contribution was showing that *ontologies* could serve as the control mechanism, replacing ad hoc configuration with formal, verifiable, reasoner-checkable structures.

Level III (2023–2024) was a qualitative jump. LLM output generation was widely treated as a stochastic process controllable only through prompt engineering—an informal, empirical, non-verifiable practice. OntoChatGPT [19] demonstrated that the same ontological control principle that governed deterministic systems could govern a fundamentally probabilistic one. The key insight was that the ontology does not need to eliminate stochasticity—it constrains the *space* within which stochastic generation occurs.

Level V (this paper) applies the same logic one step further. The RLHF preference collection process—which human corrections count as valid training data—has been treated as a matter of annotation protocol design, quality filtering heuristics, and inter-annotator agreement metrics. None of these are formal in the description logic sense: they cannot be verified by a reasoner, they do not support subsumption queries, and they do not compose into larger knowledge bases.

The domain constitution makes this process formally controllable. The five axioms (Section 3.2) define a concept ValidOversight that an OWL reasoner can evaluate automatically. This is not a metaphorical application of ontological control—it is a literal one: the same reasoning infrastructure (TBox, ABox, tableau algorithms) that classifies system architectures in Level I now classifies preference data pipelines in Level V.

The implication is that the ontological control principle is more general than any of its individual applications. It is not specifically about system architecture, NLP, or LLM alignment. It is about applying formal, verifiable, machine-checkable structure to processes that are otherwise governed by informal heuristics. The consistent success across five levels suggests that the principle's scope is

bounded by the availability of formalizable domain knowledge, not by the nature of the controlled process.

7.2 Evolutionary Cybernetics and the Stability of Oversight

Palagin et al. [22] introduced a framework for analyzing systems where goals, constraints, and structures co-evolve—a departure from classical control theory, which assumes a fixed objective function. The domain constitution operates in precisely such a regime.

Consider the feedback loop formalized in Section 4.4: the practitioner corrects the agent's output, the constitution validates the corrections, valid corrections train the model via DPO, and the improved model produces output that the practitioner then corrects differently. Each cycle potentially changes three elements simultaneously:

1. **The agent's behavior** changes because the model has been updated.
2. **The practitioner's corrections** change because the agent's output is now closer to (or further from) what the practitioner expects.
3. **The conditions for valid oversight** may change because the information asymmetry (C4) shifts as the model improves.

This is an evolutionary dynamics problem, not a static optimization problem. The domain constitution as formalized in Section 3 is a *snapshot*—it captures the conditions under which oversight is valid at a given point in the co-evolution of practitioner and agent.

Palagin et al. [22] provides the theoretical vocabulary for analyzing this dynamic. In the framework of evolutionary cybernetics, the domain constitution is an *invariant structure*—a set of constraints that must be preserved across evolutionary steps for the system to maintain its functional integrity. The question is whether the five conditions (C1–C5) are robust invariants or whether they degrade as the system evolves.

We can analyze each condition's evolutionary stability:

C1 (Shared Persistent State): Stable. The requirement for shared state does not depend on the agent's capability level. Whether the agent is weak (requiring heavy correction) or strong (requiring light correction), the shared codebase remains necessary for contextual oversight.

C2 (Compositional Layering): Stable. Compositional task structure is a property of the work domain (software engineering, legal analysis), not of the agent's capability. As long as the domain requires multi-step, interdependent work, C2 holds.

C3 (Grounding in Observable Reality): Stable. Observable success criteria are defined by the production environment, not by the human-agent interaction. Deployment failures, latency spikes, and user churn remain observable regardless of model capability.

C4 (Information Asymmetry): Potentially Unstable. As models improve, the information gap between practitioner and agent may narrow. A sufficiently capable agent with access to business context, regulatory databases, and user feedback channels might satisfy C4 only marginally. In the limit, if the agent knows everything the practitioner knows, corrections become redundant—oversight degenerates into rubber-stamping.

This is the critical evolutionary pressure on the domain constitution. C4 is the condition most likely to degrade under capability scaling, and its degradation would undermine the validity of the entire framework. Burns et al. [4] analyze a related phenomenon (weak-to-strong generalization),

where the supervisor's signal quality degrades as the supervised model approaches the supervisor's capability.

C5 (Consequential Grounding): Stable. Deployment consequences are external to the human-agent system. Customer satisfaction, revenue, and regulatory compliance do not depend on who (human or agent) produced the artifact.

The analysis yields a specific prediction: **the domain constitution is evolutionarily stable under capability scaling in 4 of 5 conditions, with C4 as the critical vulnerability.** Monitoring the information asymmetry between practitioner and agent—and detecting when it falls below a threshold sufficient for meaningful oversight—is the key challenge for maintaining valid oversight as LLM capabilities increase.

This connects directly to the scalable oversight research program [3]: the question "can humans oversee superhuman AI systems?" is, in our formalization, the question "does C4 remain satisfiable as agent capability grows?" The ontological formalization does not answer this question, but it makes it *precise*: C4 degrades when PrivateInfo (Definition 3.1, Eq. 2) approaches the empty set.

7.3 Practical Implications for RLHF Methodology

The formalization developed in this paper has three practical implications for RLHF preference data collection and curation.

Implication 1: Preference data should carry provenance metadata. Current RLHF datasets (OpenAssistant, Anthropic-HH, UltraFeedback) contain preference labels without workflow provenance: there is no metadata indicating whether the annotator operated within a persistent workflow, whether tasks composed, or whether outcomes were tracked. The domain constitution provides a minimal provenance schema: for each preference pair, record which of the five conditions were satisfied during annotation. This does not require OWL reasoning at annotation time—a simple checklist of five binary features per pair suffices to enable post-hoc filtering.

Implication 2: Oversight grade enables weighted training. The graded classification (Definition 3.3) provides a principled weighting scheme for DPO training. Rather than treating all preference pairs equally, pairs from FullOversight workflows receive weight 1.0, pairs from PartialOversight receive a discounted weight, and pairs from InvalidOversight are excluded. This is analogous to how curriculum learning prioritizes higher-quality training examples, but with the quality criterion derived from formal axioms rather than heuristic filtering.

The weighting scheme is compatible with the standard DPO objective [23]. For a preference pair (x, y_w, y_l) with oversight grade γ:

$$\mathcal{L}_{\text{weighted-DPO}} = -\mathbb{E}_{(x,y_w,y_l)}\left[w(\gamma) \cdot \log \sigma\left(\beta \log \frac{\pi_\theta(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \beta \log \frac{\pi_\theta(y_l|x)}{\pi_{\text{ref}}(y_l|x)}\right)\right] \tag{31}$$

where $w(\gamma)$ maps oversight grade to training weight. The simplest instantiation is $w(5) = 1.0$, $w(3{-}4) = \alpha$, $w(\leq 2) = 0$, where α is a hyperparameter.

Implication 3: OWL reasoning as a data pipeline component. The OWL ontology (Section 5) can be deployed as an automated filter in a preference data pipeline. Workflow metadata (session persistence, task dependencies, outcome tracking) is asserted as ABox individuals. The

HermiT reasoner classifies each workflow instance. Only instances classified as ValidOversight or PartialOversight pass to the DPO training stage.

This is architecturally lightweight: OWL reasoning over small ABoxes (thousands of workflow instances, not millions of triples) completes in sub-second time. The overhead of adding ontological filtering to a preference data pipeline is negligible compared to the cost of DPO training itself.

7.4 Limitations of the Formalization

The formalization developed here has three limitations that should be acknowledged.

Open-world assumption vs. closed-world data. OWL reasoning operates under the open-world assumption: an unstated fact is not assumed false. In practice, workflow metadata is generated by instrumentation systems that operate under the closed-world assumption: if a session dependency is not recorded, it does not exist. This mismatch means that OWL reasoning will systematically *underclassify*—workflows with incomplete metadata will fail conditions that they may actually satisfy. The monotonicity property (Proposition 3.4) mitigates this: adding metadata can only increase the oversight grade, so underclassification is conservative (false negatives, not false positives).

C4 is difficult to operationalize. Condition C4 (information asymmetry) requires that human corrections be based on information inaccessible to the agent. In the OWL formalization, this is expressed as $\exists \mathsf{basedOn}.\mathsf{PrivateInfo}$, where $\mathsf{PrivateInfo} \equiv \mathsf{Information} \sqcap \neg\exists \mathsf{accessibleTo}.\mathsf{Agent}$. In practice, determining whether a specific correction was based on private information requires either self-report by the practitioner or inference from behavioral context (e.g., the practitioner consulted an external source before making the correction). Neither method is perfectly reliable. This makes C4 the weakest condition operationally, in addition to being the least stable evolutionarily (Section 7.2).

Single-practitioner validation. The empirical validation (Section 6) uses data from a single practitioner. The formal model itself is domain-independent—the TBox makes no assumptions about the practitioner's identity, domain, or skill level. But the *instantiation* (ABox) and the *empirical claims* about oversight quality are grounded in one case study. Multi-practitioner validation is required before the formalization can be recommended as a standard component of RLHF data pipelines.

7.5 Relationship to Other Formal Approaches

Three lines of work are related to the formalization presented here but differ in scope or method.

Constitutional AI [2]. Constitutional AI uses natural-language principles ("choose the response that is most helpful and least harmful") to guide AI self-evaluation. The domain constitution differs in three ways: (a) the conditions are formal axioms, not natural language; (b) the conditions govern the *oversight process*, not the model's output; (c) the conditions are evaluated by a reasoner, not by the model itself. Constitutional AI and the domain constitution are complementary: the former structures *what to evaluate*, the latter structures *whose evaluation to trust*.

Scalable oversight [3, 9, 12]. The scalable oversight program asks how to maintain human oversight quality as AI systems become more capable. Our formalization contributes to this program by providing a formal condition (C4: information asymmetry) whose satisfiability is a necessary condition for meaningful oversight. The prediction that C4 is the critical vulnerability under

capability scaling (Section 7.2) can be tested empirically: track |PrivateInfo| over successive model generations and measure whether it converges to zero.

Ontology-based data quality [21]. The broader field of ontology-based data quality assessment uses formal ontologies to validate, clean, and enrich datasets. Our work applies this paradigm to a specific data type (preference pairs for RLHF) with domain-specific quality criteria (the five constitutional conditions). The contribution relative to general ontology-based data quality is the *content* of the quality criteria, not the *method* of applying them.

8 Conclusion

We have extended the principle of ontology-controlled systems [16] from the control of system output to the control of human oversight over system output. The domain constitution—five axioms in $\mathcal{SHOIQ}$ description logic—provides formal, decidable, machine-checkable criteria for determining when human corrections on LLM-agentic output constitute valid training signal for RLHF.

The formalization yields three results. First, the five conditions are formally independent: no condition is entailed by the conjunction of the remaining four (Proposition 3.3). This confirms that each condition captures a distinct aspect of oversight validity that cannot be derived from the others. Second, ontological control of human oversight (ValidOversight) is a strict specialization of ontological control of LLM output (OntoChatGPT_Control): every valid oversight instance satisfies the conditions for ontology-controlled output, but not conversely (Proposition 4.1). This formally confirms that edit-trace oversight extends the ontology-controlled paradigm, inheriting its foundation (C1, C3) and adding oversight-specific conditions (C2, C4, C5). Their integration into a single system is satisfiable (Theorem 4.1). Third, among the five conditions, C4 (information asymmetry) is identified as the critical evolutionary vulnerability: it is the only condition whose satisfiability depends on the agent's capability level, connecting the formalization to the scalable oversight research program [3].

The OWL 2 DL ontology implementing the domain constitution is available for automated reasoning. Given workflow metadata as ABox assertions, a standard OWL reasoner (HermiT [6], Pellet [25]) classifies each workflow instance as full, partial, or invalid oversight in sub-second time. This enables ontology-based filtering of RLHF preference data as a lightweight, formally grounded pipeline component.

The principal limitation is empirical: the formalization has been validated on a single practitioner's data. The formal model is domain-independent, but its practical value depends on multi-practitioner validation across diverse domains. This validation, together with the implementation of the weighted DPO training objective, is the subject of ongoing work.

References

[1] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 1st edition, 2003.

[2] Yuntao Bai, Saurav Kadavath, Sandipan Kundu, et al. Constitutional AI: Harmlessness from AI feedback. *arXiv preprint arXiv:2212.08073*, 2022.

[3] Samuel R. Bowman et al. Measuring progress on scalable oversight for large language models. *arXiv preprint arXiv:2211.03540*, 2022.

[4] Collin Burns et al. Weak-to-strong generalization: Eliciting strong capabilities with weak supervision. *arXiv preprint arXiv:2312.09390*, 2023.

[5] Paul F. Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

[6] Birte Glimm, Ian Horrocks, Boris Motik, Giorgos Stoilos, and Zhe Wang. Hermit: An OWL 2 reasoner. *Journal of Automated Reasoning*, 53(3):245–269, 2014. doi: 10.1007/s10817-014-9305-1.

[7] Bernardo Cuenca Grau, Ian Horrocks, Boris Motik, Bijan Parsia, Peter Patel-Schneider, and Ulrike Sattler. OWL 2: The next step for OWL. *Journal of Web Semantics*, 6(4):309–322, 2008. doi: 10.1016/j.websem.2008.05.001.

[8] Ian Horrocks, Oliver Kutz, and Ulrike Sattler. Even more irresistible SROIQ. *Proceedings of the 10th International Conference on Principles of Knowledge Representation and Reasoning (KR)*, pages 57–67, 2006.

[9] Geoffrey Irving, Paul Christiano, and Dario Amodei. AI safety via debate. *arXiv preprint arXiv:1805.00899*, 2018.

[10] Jean-Baptiste Lamy. Owlready: Ontology-oriented programming in Python with automatic classification and high level constructs for biomedical ontologies. *Artificial Intelligence in Medicine*, 80:11–28, 2017. doi: 10.1016/j.artmed.2017.07.002.

[11] Harrison Lee, Samrat Phatale, Hassan Mansoor, et al. RLAIF: Scaling reinforcement learning from human feedback with AI feedback. In *International Conference on Learning Representations*, 2024.

[12] Jan Leike, David Krueger, Tom Everitt, Miljan Martic, Vishal Maini, and Shane Legg. Scalable agent alignment via reward modeling: A research direction. *arXiv preprint arXiv:1811.07871*, 2018.

[13] Anna Litvin, Oleksandr Palagin, Vladislav Kaverinskiy, and Kyrylo Malakhov. Ontology-driven development of dialogue systems. *South African Computer Journal*, 35(1), 2023. doi: 10.18489/sacj.v35i1.1233.

[14] Long Ouyang, Jeffrey Wu, Xu Jiang, et al. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, volume 35, 2022.

[15] Vladimir Ovcharov. Edit-trace oversight: Scalable alignment signal from agentic workflows. *arXiv preprint*, 2026. Under review.

[16] Alexander V. Palagin. Architecture of ontology-controlled computer systems. *Cybernetics and Systems Analysis*, 42(2):254–264, 2006. doi: 10.1007/s10559-006-0061-z.

[17] Alexander V. Palagin, Serhiy L. Kryvyi, and Mykola G. Petrenko. On the automation of the process of extracting knowledge from natural language texts. *Natural and Artificial Intelligence International Book Series*, 2012. Bibliographic details incomplete.

[18] Oleksandr Palagin, Vitalii Velychko, Kyrylo Malakhov, and Borys Shchurov. Distributional semantic modeling: A revised technique to train term/word vector space models applying the ontology-related approach. *arXiv preprint arXiv:2003.03350*, 2020. URL https://arxiv.org/abs/2003.03350.

[19] Oleksandr Palagin, Vladislav Kaverinskiy, Anna Litvin, and Kyrylo Malakhov. OntoChatGPT information system: Ontology-driven structured prompts for ChatGPT meta-learning. *International Journal of Computing*, 22(2):170–183, 2023. URL https://arxiv.org/abs/2307.05082.

[20] Oleksandr Palagin, Vladislav Kaverinskiy, and Kyrylo Malakhov. Fundamentals of the integrated use of neural network and ontolinguistic paradigms: A comprehensive approach. *Cybernetics and Systems Analysis*, 60:111–123, 2024. doi: 10.1007/s10559-024-00652-z.

[21] Oleksandr Palagin, Mykola Petrenko, and Kyrylo Malakhov. Challenges and role of ontology engineering in creating the knowledge industry. *Cybernetics and Systems Analysis*, 60:633–645, 2024. doi: 10.1007/s10559-024-00702-6.

[22] Oleksandr Palagin, Dmytro Symonov, and Mykhailo Chervynskyi. Modelling evolutionary cybernetics: Ontology, invariants and design principles. *Information Technologies and Systems*, 6(6):3–29, 2025. doi: 10.15407/intechsys.2025.06.003.

[23] Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

[24] Rob Shearer, Boris Motik, and Ian Horrocks. HermiT: A highly-efficient OWL reasoner. *Proceedings of the 5th International Workshop on OWL: Experiences and Directions (OWLED)*, 2008.

[25] Evren Sirin, Bijan Parsia, Bernardo Cuenca Grau, Aditya Kalyanpur, and Yarden Katz. Pellet: A practical OWL-DL reasoner. *Journal of Web Semantics*, 5(2):51–53, 2007. doi: 10.1016/j.websem.2007.03.004.