HyDRA: A Hybrid-Driven Reasoning Architecture for Verifiable Knowledge Graphs

Adrian Kaiser[1,*], Claudiu Leoveanu-Condrei[1], Ryan Gold[1], Marius-Constantin Dinu[1,2] and Markus Hofmarcher[1]

[1]*ExtensityAI, Austria*
[2]*WU Executive Academy, Austria*

Abstract
The synergy between symbolic knowledge, often represented by Knowledge Graphs (KGs), and the generative capabilities of neural networks is central to advancing neurosymbolic AI. A primary bottleneck in realizing this potential is the difficulty of automating KG construction, which faces challenges related to output reliability, consistency, and verifiability. These issues can manifest as structural inconsistencies within the generated graphs, such as the formation of disconnected *isolated islands* of data or the inaccurate conflation of abstract classes with specific instances. To address these challenges, we propose HyDRA, a **Hy**brid-**D**riven **R**easoning **A**rchitecture designed for verifiable KG automation. Given a domain or an initial set of documents, HyDRA first constructs an ontology via a panel of collaborative neurosymbolic agents. These agents collaboratively agree on a set of competency questions (CQs) that define the scope and requirements the ontology must be able to answer. Given these CQs, we build an ontology graph that subsequently guides the automated extraction of triplets for KG generation from arbitrary documents. Inspired by design-by-contracts (DbC) principles, our method leverages verifiable contracts as the primary control mechanism to steer the generative process of Large Language Models (LLMs). To verify the output of our approach, we extend beyond standard benchmarks and propose an evaluation framework that assesses the functional correctness of the resulting KG by leveraging symbolic verifications as described by the neurosymbolic AI framework, *SymbolicAI*. This work contributes a hybrid-driven architecture for improving the reliability of automated KG construction and the exploration of evaluation methods for measuring the functional integrity of its output. The code is publicly available.

Keywords
ontology, knowledge graph, neurosymbolic, design-by-contract, large language models

1. Introduction

The advancement of neurosymbolic AI hinges on the effective fusion of sub-symbolic pattern recognition, characteristic of neural networks, and the structured, explicit reasoning enabled by symbolic knowledge representations [1, 2, 3, 4]. As such, Knowledge Graphs (KGs) are a core component of the symbolic approach, offering a powerful formalism for encoding information. The prospect of using Large Language Models (LLMs) to automate KG construction promises to bridge these two worlds [5, 6, 7], potentially augmenting how intelligent systems acquire and reason over knowledge [1, 8].

However, reliably constructing a KG from a given dataset is still an unsolved topic [9, 10, 11]. Unconstrained, LLMs generate KGs with structural and semantic flaws that undermine their utility. Outputs are often inconsistent, containing disconnected subgraphs—or *isolated islands*—that break the KG's conceptual integrity, or conflating abstract classes with concrete instances in ways that violate ontological principles or capture KG semantics [12].

These issues necessitate time-consuming manual validation and correction, defeating the purpose of automation and creating a significant barrier to deploying trustworthy, KG-augmented AI systems.

One of the core issues with KG construction from unstructured data is the problem of *entity resolution* (ER) [13]. In the context of automated KG generation from unstructured input ER refers to the challenge of deciding when and how to consolidate entities or concepts that may refer to the same real-world object but appear differently in the data. This problem is central to ensuring the consistency and accuracy of the generated KG, especially when the KG is constructed iteratively, i.e. from one set of documents at a time.

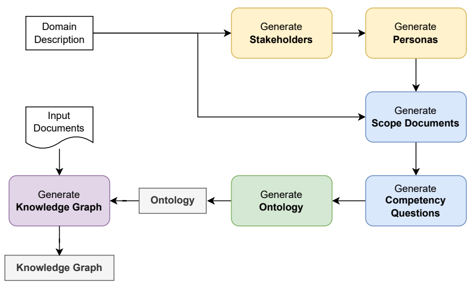


Figure 1: Overview of the HyDRA architecture. The ontology generation process begins with a user-defined domain description. HyDRA first generates stakeholder groups, which define the constraints for generating personas (yellow). It then creates *scope documents* and competency questions (blue), which are used to construct an ontology (green). If input documents are supplied, HyDRA can, as an optional final step, generate a knowledge graph (purple) from a set of documents adhering to the constructed ontology.

To address this, we leverage a recent paradigm shift following neurosymbolic architectures [14, 15]: treating the construction of KG not as an unconstrained generative task, but as a verifiable multistep generative process. We introduce HyDRA [1] (a **Hy**brid-**D**riven **R**easoning **A**rchitecture), a system that operationalizes this idea through the principles of Design-by-Contract (DbC) [16] and neurosymbolic architecture [14]. HyDRA orchestrates an LLM-driven pipeline using a series of verifiable contracts, where each stage of construction, from requirements gathering to instance population, is governed by explicit, machine-readable specifications. Our architecture implements a neurosymbolic

ANSyA 2025: 1st International Workshop on Advanced Neuro-Symbolic Applications, co-located with ECAI 2025.
*Corresponding author.
adrian.kaiser@extensity.ai (A. Kaiser)
© 2025 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

[1]Codebase: https://github.com/ExtensityAI/ontology-hydra

feedback loop in which the symbolic layer (the contract) actively steers and corrects the generative output of the neural layer (the LLM).

The process begins by defining a *scope contract* through the generation of competency questions (CQ), which then serve as formal preconditions for creating the *ontology schema*. This schema, governed by its own *schema contract*, becomes a precondition for the final KG population. This creates an end-to-end contract lineage, ensuring traceability from high-level requirements to low-level data. If a contract's postcondition is violated at any stage, HyDRA's core innovation—a closed-loop verification and repair mechanism—is triggered, automatically re-prompting the LLM or executing corrective actions until the KG is verifiably compliant.

In order to test the quality of the HyDRA constructed KG, we selected a challenging question-answering benchmark from the medical domain. We selected question-answering as the initial benchmark as datasets of this kind are readily available, Since questions and answers are initially unstructured data itself, we indirectly evaluate the quality of the KG by supplying the KG in the context of an LLM and then query for the answers. Surprisingly, unconstrained KG construction (i.e. without the constraint of an ontology) performs better on such benchmarks. We identify the underlying reason and propose a novel evaluation scheme to evaluate KG quality. Our goal is to establish a benchmark that directly measures the quality of the graph structure by using graph databases such as Neo4j and converting Q&A pairs into queries that can be interpreted by such databases. However, this poses a number of technical and conceptual challenges that we had to address, and we are thus still in the process of implementing this procedure. In summary, this work makes the following contributions:

- A hybrid-driven architecture, HyDRA, that adapts Design-by-Contract principles to orchestrate and verify LLM-driven KG construction.
- A **closed loop contract-driven repair mechanism** that automatically enforces structural invariants, such as graph connectivity and type consistency, through iterative refinement.
- A concept for an **evaluation framework** to measure the practical utility of a KG by leveraging neurosymbolic design principles.

2. Related Work

This section situates HyDRA within the context of prior research, focusing on four key areas. We associate neurosymbolic approaches with KG methodologies and articulate how HyDRA's contract-centric architecture presents a novel solution.

2.1. Automated Knowledge-Graph (ABox) Construction

Systems for populating KGs with instance data (ABoxes) range from classic NLP pipelines like *FRED* [17] to modern LLM-driven frameworks like *GraphRAG* [18] and *AdaKGC* [19]. While these tools demonstrate high accuracy in extracting individual triplets, their validation capabilities are often confined to local constraints, such as checking a relation's arity or a triplet's adherence to a basic type. They typically lack mechanisms to enforce global structural invariants across the entire graph.

HyDRA's primary differentiation lies in its introduction of a **principled, contract-driven verification loop to ensure global structural integrity**. Using standards like the Shapes Constraint Language (SHACL) [20] or graph-based queries, HyDRA defines postcondition contracts that check for global properties, such as the absence of isolated components or the prevention of class-instance conflation. Its iterative repair mechanism systematically identifies and rectifies these structural violations, producing a graph that is not only rich in facts but also structurally sound.

2.2. LLMs for Ontology (TBox) Generation

The automated generation of ontologies (TBoxes) with LLMs has shown significant promise. Frameworks like *LLMs4OL* [21] and *OLLM* [22] demonstrate the ability of LLMs to generate taxonomic and relational structures from text using prompting and fine-tuning. These methods excel at generating candidate schema components with high recall. The common limitation, however, lies in their validation strategies, which are often separated from the generation process. Validation frequently relies on after-the-fact checks, heuristic filtering, or indispensable human-in-the-loop curation to ensure quality and correctness.

HyDRA addresses this gap by tightly **coupling generation with in-loop verification**. It introduces machine-checkable contracts—such as ensuring a schema is valid OWL-DL or that all classes possess natural language descriptions—as formal postconditions of the generation step. A contract violation is not merely a terminal failure but a signal that triggers a corrective re-prompting or refinement action. This integration moves ontology generation away from a *generate-then-validate* model towards a more reliable, correct-by-construction approach.

2.3. Competency-Question (CQ) Engineering

Since their formalization by Gruninger and Fox [23], Competency Questions (CQs) have been central to defining an ontology's scope. The manual effort of creating CQs is now being alleviated by LLM-based automation, with techniques like Retrieval-Augmented Generation (RAG) for domain grounding [24] and multi-persona prompting for diverse perspectives [25]. While these methods improve the quality and coverage of CQs, they still treat them primarily as informal, human-readable documentation that guides the subsequent development process.

HyDRA fundamentally changes the role of CQs by **elevating them to formal, machine-verifiable preconditions**. This is a two-fold innovation. First, the set of CQs becomes the root *scope contract* that the entire downstream pipeline must satisfy. Second, HyDRA establishes a **verifiable lineage** by embedding CQ identifiers within the generated ontology's metadata. This unique feature allows any schema component to be programmatically traced back to the specific requirement that motivated its existence, providing an unprecedented level of transparency and auditability.

2.4. Ontology and Knowledge-Graph Evaluation

The limitations of traditional static KG evaluation metrics are well documented [26], leading to a call for more functional, task-oriented evaluation methodologies [27]. Modern approaches have begun to address this by evaluating KGs based on their utility in downstream tasks (*KGrEaT* [28]) or by developing methods to translate natural language questions into formal queries (*GraphRAFT* [29]).

HyDRA's evaluation framework represents the logical conclusion of this trend, creating a perfectly symmetrical and closed-loop process. The very CQs that serve as the initial requirement contract are programmatically translated into executable queries for the final artifact. Specifically, we query the generated graph via verifiable operators using neurosymbolic design principles. The success rate of these queries provides a direct measure of the KG's functional correctness. This approach ensures that the KG is not just structurally valid but useful for the exact purpose it was designed to serve.

3. Background

3.1. Design-by-Contract

Design-by-Contract (DbC) is a software engineering methodology introduced by Bertrand Meyer [30, 16] that emphasizes formal specifications for software components. DbC treats the relationship between a component and its clients as a formal agreement, similar to a legal contract, where each party has obligations and benefits. This methodology is built on three key elements: preconditions (obligations on the caller), postconditions (obligations on the component), and invariants (conditions that must always hold). DbC enhances software reliability by making explicit the assumptions and guarantees of each component, enabling systematic verification and validation throughout the development lifecycle. This paradigm offers the theoretical blueprint for HyDRA.

In recent years, researchers have begun adapting these concepts to AI systems. This has led to probabilistic contracts that can handle model uncertainty [31], assume-guarantee reasoning frameworks for compositional verification like *Pacti* [32], and runtime enforcement tools like *MLGuard* [33] and *NVIDIA NeMo Guardrails* [34] that act as safety-oriented wrappers for ML/LLM systems.

HyDRA adapts these DbC principles to KG construction, providing a framework for ensuring correctness, consistency, and completeness throughout the KG development process. By formalizing the requirements and expectations at each stage, DbC principles enable HyDRA to establish verifiable pathways from conceptual models to implemented knowledge structures.

3.2. Definition of Contracts

A contract in this context comprises four key components: preconditions, postconditions, invariants, and a verification function. Preconditions and postconditions are analogous to the general definition, and invariants establish properties that remain constant throughout the operation. The verification function determines whether these conditions are satisfied and is realized via neurosymbolic operations.

These contracts form a composition of operations in HyDRA, where the postcondition of one phase becomes the precondition of the next [35, 36]. This creates a verifiable pathway from initial requirements to final implementation, ensuring traceability and correctness at each step of the KG construction process. When a contract's conditions are violated, HyDRA's verification engine triggers remediation actions to restore compliance, establishing a self-correcting mechanism that maintains the integrity of the KG throughout its development lifecycle.

3.3. SymbolicAI Framework

The SymbolicAI framework [14] provides the theoretical and technical foundation for HyDRA's neurosymbolic approach. At its core, SymbolicAI treats LLMs as semantic parsers. The framework introduces polymorphic, compositional, and self-referential operations for multi-modal data that connects multi-step generative processes. SymbolicAI implements a contract-based validation layer that enforces preconditions, postconditions, and type validation using structured data models, with automatic remediation through validation functions and retry logic. This allows HyDRA to take advantage of the pattern recognition capabilities of neural networks (e.g., LLMs) while maintaining the explainability and verifiability of symbolic layers (e.g., contracts).

4. HyDRA Architecture

HyDRA provides an end-to-end framework for building domain-specific ontologies and KGs through an iterative, simulated stakeholder-driven process. HyDRA's architecture is illustrated in Figure 1. The process begins with a user-defined domain description and some input documents that are used to populate the KG. First, stakeholder groups are generated from the domain description, followed by persona generation for each group. These groups are used to produce so-called scope documents which define the conceptual boundaries of the domain. The scope documents are used to generate competency questions which guide the iterative construction of the ontology. Finally, the resulting ontology and the input documents are used to generate a KG.

Each phase builds on the outputs of the previous steps, ensuring that the final KG is relevant and aligned with the domain. HyDRA does not rely on the input documents for ontology generation, thereby avoiding fitting to the initial, potentially limited sample. Instead, it aims to capture a more general representation of the target domain.

4.1. Persona Generation

Instead of directly generating personas, we first prompt the LLM to produce a prioritized list of stakeholder groups relevant to the domain. For each group, distinct personas are generated, with the number of personas per group increasing with the group's priority. This approach ensures that more influential or central groups are represented by a broader set of perspectives. After generation, all personas are aggregated, shuffled, and divided into n heterogeneous groups G_i, where n is set as the number of aggregated personas divided by group size l (with $l = 4$ as default hyperparameter). These persona groups are then used as inputs for scope document generation in the next step.

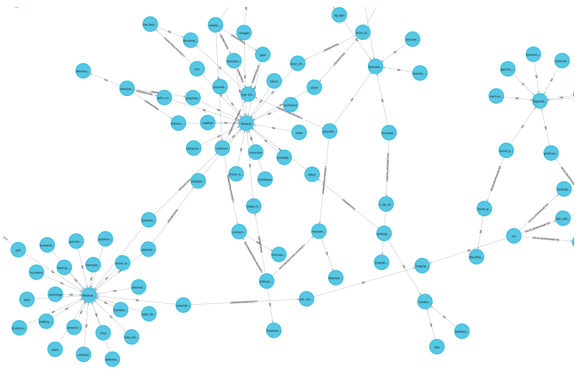


Figure 2: HyDRA-generated, ontology-conditioned KG inserted into Neo4j as one-to-one mapping (partial graph is pictured).

4.2. Scope Generation

Given an initial natural-language domain description d provided by the user and the set of heterogenous persona groups identified in the previous phase, HyDRA generates scope documents that define both the conceptual boundaries and the core area of the domain. These scope documents subsequently serve as references for competency question generation. For each persona group G_i, one scope document S_i is generated using the LLM and d, resulting in an initial set of n scope documents. The scope documents are then recursively merged in batches of size k (with $k = 6$ as the default hyperparameter) by prompting an LLM to merge the current batch of documents until a single, consolidated scope document remains. The complete process is outlined in Algorithm 1.

Algorithm 1 Scope Document Generation

```
1: Input: Domain description $d$, persona groups $\mathcal{G} = \{G_1, G_2, \ldots, G_n\}$, batch size $k$
2: $\mathcal{S} \leftarrow \emptyset$
3: for each persona group $G_i$ in $\mathcal{G}$ do
4:     $S_i \leftarrow \mathrm{LLM}(d, G_i, \mathrm{prompt}_{\mathrm{scope}})$
5:     $\mathcal{S} \leftarrow \mathcal{S} \cup \{S_i\}$
6: end for
7: while $|\mathcal{S}| > 1$ do
8:     Partition $\mathcal{S}$ into batches of size $k$ (default $k = 6$)
9:     $\mathcal{S}_{\mathrm{new}} \leftarrow \emptyset$
10:    for each batch $B$ in partition do
11:        $S_{\mathrm{merged}} \leftarrow \mathrm{LLM}(d, B, \mathrm{prompt}_{\mathrm{merge}})$
12:        $\mathcal{S}_{\mathrm{new}} \leftarrow \mathcal{S}_{\mathrm{new}} \cup \{S_{\mathrm{merged}}\}$
13:    end for
14:    $\mathcal{S} \leftarrow \mathcal{S}_{\mathrm{new}}$
15: end while
16: return the final scope document in $\mathcal{S}$
```

4.3. Competency Question Generation

Taking the previously generated persona groups G_i and the final scope document, HyDRA generates n batches of competency questions that are then merged and deduplicated via a simple LLM-based aggregation step. The competency questions are then passed to the ontology generator.

4.4. Ontology Generation

For simplicity, we focus on a lightweight subset of OWL2—a standardized ontology language for formally representing classes, properties and relationships. Specifically, we support RDFS class hierarchies, object and data properties with domains and ranges, a fixed set of XSD datatypes (*xsd:string, xsd:integer, ...*) and selected OWL2 property characteristics (*functional, transitive, ...*). HyDRA iteratively builds the ontology based on batches of competency questions. Algorithm 2 outlines this process. The ontology $\mathcal{O}$ is updated in each iteration with new concepts C_i generated by the LLM for a batch of competency questions Q_i. The candidate concepts C_i are merged with the current ontology to form a candidate ontology $\mathcal{O}'$. If $\mathcal{O}'$ does not violate any constraints, it becomes the new ontology; otherwise, the LLM is repeatedly called to revise its output until all constraints are satisfied. The process uses two types of prompts: $\mathrm{prompt}_{\mathrm{ontology}}$ for initial ontology construction and $\mathrm{prompt}_{\mathrm{fix_o}}$ for revision in the case of constraint violations.

Algorithm 2 Ontology Generation Algorithm

```
1: Input: A set of competency questions, partitioned into batches $\mathcal{Q} = \{Q_1, Q_2, \ldots, Q_m\}$
2: $\mathcal{O} \leftarrow \emptyset$
3: for each batch of competency questions $Q_i$ do
4:     $C_i \leftarrow \mathrm{LLM}(Q_i, \mathcal{O}, \mathrm{prompt}_{\mathrm{ontology}})$
5:     $\mathcal{O}' \leftarrow \mathcal{O} \cup C_i$
6:     Violations $\leftarrow$ FindViolations$(\mathcal{O}')$
7:     while Violations $\neq \emptyset$ do
8:         $C_i \leftarrow \mathrm{LLM}(\mathrm{Violations}, C_i, \mathrm{prompt}_{\mathrm{fix_o}})$
9:         $\mathcal{O}' \leftarrow \mathcal{O} \cup C_i$
10:        Violations $\leftarrow$ FindViolations$(\mathcal{O}')$
11:    end while
12:    $\mathcal{O} \leftarrow \mathcal{O}'$
13: end for
14: return $\mathcal{O}$
```

4.4.1. Validation

The validation step ensures that each candidate ontology produced during generation satisfies a set of predefined constraints before being accepted. HyDRA applies the contract pattern to enforce these constraints after each merge of new concepts. Constraints are checked with respect to the current ontology. Thus, as the ontology grows, validation effectively becomes stricter. For clarity, we distinguish between simple constraints and complex constraints:

Simple Constraints Simple constraints are fundamental syntactic and referential checks that validate individual ontology elements in isolation. These include naming conventions (class names must start with uppercase letters), uniqueness checks (no duplicate classes or properties), existence validations (referenced classes must exist), and property domain/range validations (ensuring properties reference valid classes).

Complex Constraints Complex constraints are structural validations that ensure the ontology maintains a proper hierarchical structure. These include preventing circular inheritance relationships (where class A extends B while B extends A directly or indirectly) and prohibiting self-inheritance (a class cannot be its own superclass).

Additionally, we enforce single inheritance (each class should have exactly one superclass except the root). While OWL2 allows multi inheritance, we chose to disallow it to keep generated ontologies simple. Similarly, while any class is implicitly a subclass of *owl:Thing*, we enforce a single explicit root class to avoid disconnected hierarchies during iterative construction and to make the structure easier to validate.

Retries Whenever at least one constraint is violated during generation, the LLM is prompted with both the violations and its previous output to revise its proposal. HyDRA only accepts and merges concepts into the ontology if no violations occur, thus ensuring correctness at each step. Accepting only the non-violating subset and prompting the model to fix only the violations would restrict the LLM from restructuring its output, including changes or removals of concepts that did not initially violate constraints.

4.5. Knowledge Graph Generation

HyDRA uses the generated ontology to populate a KG from input data such as text documents. KG generation, like ontology generation, is performed iteratively and additively. HyDRA splits the input documents into chunks and extracts candidate triplets of the form (subject, predicate, object), with a special predicate *isA* for class assignments. Triplet extraction is performed over multiple epochs to allow information from different document parts to be connected. As opposed to ontology generation (where the whole candidate ontology is validated at each step), only the newly-generated triplets are validated. In Algorithm 3, the KG $\mathcal{K}$ is incrementally updated with candidate triplets T_j generated by the LLM for each document chunk D_j. The ontology $\mathcal{O}$ provides type and property constraints for validation. Two types of prompts are used: $\text{prompt}_{\text{kg}}$ for initial triplet extraction and $\text{prompt}_{\text{fix}_{\text{kg}}}$ for revision in case of violations.

Algorithm 3 Knowledge Graph Generation Algorithm

```
1:  Input: A set of document chunks $\mathcal{D} = \{D_1, D_2, \ldots, D_p\}$, an ontology $\mathcal{O}$
2:  $\mathcal{K} \leftarrow \emptyset$
3:  for each epoch do
4:      for each document chunk $D_j$ do
5:          $T_j \leftarrow \text{LLM}(D_j, \mathcal{K}, \mathcal{O}, \text{prompt}_{\text{kg}})$
6:          $\text{Violations} \leftarrow \text{FindViolations}(T_j, \mathcal{K}, \mathcal{O})$
7:          while $\text{Violations} \neq \emptyset$ do
8:              $T_j \leftarrow \text{LLM}(\text{Violations}, T_j, \text{prompt}_{\text{fix}_{\text{kg}}})$
9:              $\text{Violations} \leftarrow \text{FindViolations}(T_j, \mathcal{K}, \mathcal{O})$
10:         end while
11:         $\mathcal{K} \leftarrow \mathcal{K} \cup T_j$
12:     end for
13: end for
14: return $\mathcal{K}$
```

4.5.1. Triplet Validation

Triplet validation ensures that each candidate KG remains consistent with the ontology schema and all previously accepted triplets. HyDRA again applies both syntactic checks (e.g., duplicates, naming conventions) and semantic checks derived from the ontology. In particular, triplets are validated against property domains and ranges, datatype restrictions, and class assignments defined during ontology generation.

For example, if the ontology declares that an object property *hasAuthor* must link a *ResearchPaper* to a *Person*, but a triplet proposes *(ResearchPaper, hasAuthor, Institution)*, a violation is detected. Similarly, if a data property *publicationYear* is restricted to *xsd:integer* but the extracted value is a string (e.g., "twenty twenty-three"), the triplet is rejected.

Class Reassignment While HyDRA generally disallows reassigning an entity to a completely different class, it does permit class narrowing. If an entity is initially assigned to class A, and a later triplet assigns it to a subclass B of A, this reassignment is accepted since B inherits all properties from A. This reflects the assumption that different documents may progressively specify entity types with increasing precision.

5. Experimental Setup

Assessing the quality of a KG, and by proxy the ontology, is difficult as, to the best of our knowledge, no benchmarks exist for this purpose. Therefore, we turn to the task of question-answering for initial experiments as benchmarks are readily available and KGs are well suited for this task. While we identified several issues with this evaluation we find our results interesting as they help us to design a novel evaluation scheme,

5.1. Benchmark Configuration

We used the Biomedical Engineering subset of the MedExQA dataset [37] for evaluation due to its robust question-answer structure. The MedExQA dataset was specifically chosen for its strength in capturing structured question-answer pairs relevant across multiple domains in the medical field. While not part of this work this enables the iterative construction of a KG spanning the full domain. In this work, we explore HyDRAs capabilities on the biomedical engineering domain of MedExQA.

5.2. Benchmark Domain

We selected the Biomedical Engineering domain from the MedExQA dataset to assess HyDRA effectiveness and generalizability. The dataset contains four additional domains of interest but we restricted our experiments to a single domain due to computational constraints.

- **Biomedical Engineering (MedExQA)**: Representing biomedical devices, methodologies, diagnostic and therapeutic procedures, engineering principles, and patient care interactions.

For the benchmark, we created a concise domain-specific description derived from the Biomedical Engineering subset of MedExQA to generate an ontology. This ontology, combined with the structured question-answer data, was then used to construct a KG. We compare this approach to a baseline KG constructed solely from the structured question-answer dataset without any domain description or ontology generation. Both KGs were subsequently evaluated using OpenAI's GPT-4.1 [38] to assess their effectiveness

5.3. Evaluation Framework

We compared our ontology-enhanced KG construction approach against a baseline method utilizing only the structured question-answer data from MedExQA, without domain descriptions or ontology generation. Evaluation of both approaches was conducted using GPT-4.1. All experiments were repeated three times with different random seeds to account for the variability in the LLM output. We report the averages.

We defined and implemented a contract that takes as input the KG, the question, and the possible answers, and outputs a best guess based solely on the provided KG. A fuzzy matching algorithm is then used to detect whether an answer is correct based on the expected answer provided by the dataset.

6. Results and Discussion

6.1. Quantitative Results

Our evaluation shows that the KG constructed without an ontology achieved higher accuracy compared to the ontology-enhanced approach. Our quantitative experimental results for the 143 question-answer pairs of the Biomedical Engineering MedExQA Dataset are summarized below:

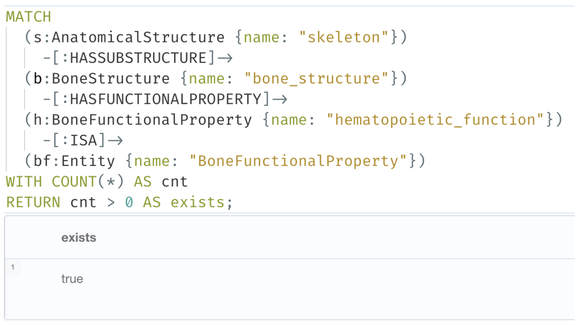
	Gemini-2.5-Flash	o4-mini	o3
Ontology Runtime	17.39	48.86	214.04
KG w/ ontology Runtime	65.35	63.41	98.10
KG w/o ontology Runtime	34.06	35.68	69.87
Evaluation Runtime	8.14	6.77	7.89
HyDRA	**57.3%**	**42.0%**	**61.5%**
Baseline	**95.1%**	**97.2%**	**97.9%**

Table 1
Performance (accuracy) and runtime metrics (in minutes) for KG construction and evaluation with and without ontology across different models.

6.2. Analysis of Benchmark-Induced Artifacts

The results presented in Table 6.1 reveal a counterintuitive outcome: the unconstrained model achieves higher performance than its ontology-guided counterpart. A qualitative analysis of the evaluation data suggests that this is a consequence of the benchmark's inherent simplicity rather than a failure of the ontology-guided method.

We found that most queries in the benchmark are solvable via a single relational hop. Such relations are directly extracted as triplets during the construction of the unconstrained KG. Because this process connects nodes from the document corpora without strict semantic constraints, the LLM can often answer these queries through a straightforward retrieval mechanism. This evaluation scheme, therefore, fails to test for the very scenarios where an ontology provides a distinct advantage: tasks that demand multi-hop inference to construct complex dependencies between entities. A benchmark with a reliable methodology for assessing such multi-step reasoning would be required to demonstrate the benefits of an ontological structure.



```
MATCH
  (s:AnatomicalStructure {name: "skeleton"})
    -[:HASSUBSTRUCTURE]→
  (b:BoneStructure {name: "bone_structure"})
    -[:HASFUNCTIONALPROPERTY]→
  (h:BoneFunctionalProperty {name: "hematopoietic_function"})
    -[:ISA]→
  (bf:Entity {name: "BoneFunctionalProperty"})
WITH COUNT(*) AS cnt
RETURN cnt > 0 AS exists;
```

```
     exists
1    true
```

Figure 3: We use the generated Neo4j Cypher query to directly query the knowledge graph that has been inserted into a Neo4j database. The chain of relationships exists, so we obtain a boolean `true` value. This Cypher query translates to the natural language question: "Is the skeleton composed of a bone structure that has the functional property *hematopoietic_function*, and is that *hematopoietic_function* classified as a *BoneFunctionalProperty*?"

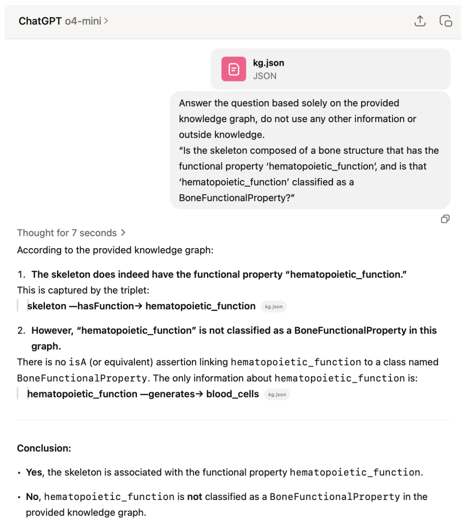
To illustrate this, we generate 3 prepositional logic statements in both natural language and converted to Neo4j cypher queries. Answering these statements requires at least 2 or 3 node traversals.

- Is the skeleton composed of a bone structure that has the functional property *hematopoietic_function*, and is that *hematopoietic_function* classified as a *BoneFunctionalProperty*?
- Does the patient monitoring device measure body temperature, and is that body temperature classified as a physiological parameter?
- Does the artificial blood vessel use teflon as a material, and is teflon classified as a Material?

Figure 3 shows one of these cypher queries and its results as an example. The query can only be resolved by following several edges through the knowledge graph. Figure 4 shows the results of using the same knowledge graph provided as context to an LLM and prompted with the natural language version of the query. All three questions can be correctly answered by simple cypher queries but LLMs fail to correctly answer these statements.

6.3. Discussion

These results highlight a clear trade-off between flexibility and exactness, and structural consistency. The higher accuracy observed in the non-ontology approach can be attributed to greater freedom and exactness in generating subject, predicate, object triplets from the question-answer data. Conversely, the ontology-based method intentionally restricts the knowledge representation to maintain consistency and shared structure. This becomes critical when integrating knowledge from multiple diverse sources and doing higher level reasoning on KGs, which isn't necessarily represented by the questions in the MedExQA dataset. The structured nature provided by the ontology ensures greater compatibility and consistency across various knowledge sources, but restricts the amount of nodes and relationships generated and the exactness in regards to the original data.


ChatGPT o4-mini >

kg.json
JSON

Answer the question based solely on the provided knowledge graph, do not use any other information or outside knowledge.
"Is the skeleton composed of a bone structure that has the functional property 'hematopoietic_function', and is that 'hematopoietic_function' classified as a BoneFunctionalProperty?"

Thought for 7 seconds >

According to the provided knowledge graph:

1. **The skeleton does indeed have the functional property "hematopoietic_function."**

This is captured by the triplet:

> **skeleton —hasFunction→ hematopoietic_function** kg.json

2. **However, "hematopoietic_function" is not classified as a BoneFunctionalProperty in this graph.**

There is no `isA` (or equivalent) assertion linking `hematopoietic_function` to a class named `BoneFunctionalProperty`. The only information about `hematopoietic_function` is:

> **hematopoietic_function —generates→ blood_cells** kg.json

Conclusion:

- **Yes**, the skeleton is associated with the functional property `hematopoietic_function`.
- **No**, `hematopoietic_function` is **not** classified as a `BoneFunctionalProperty` in the provided knowledge graph.

Figure 4: We prompt OpenAI's o4-mini with the natural language version of the cypher query, given the same knowledge graph. The model answers that the first relationship exists, but cannot validate the second condition.

We believe that the observed performance gap is mainly caused by the restrictive nature of HyDRA's ontology. This effect is especially prominent in question-answering benchmarks when the questions can be answered by identifying the information of a single node, as is often the case. However, once we add iterative KG construction and more complex queries, we believe that our approach will perform much better than the baseline.

Additionally, we identified the LLM-based evaluation as problematic, as it is impossible to distinguish between the models knowledge/capabilities and the quality of the extracted graph. Therefore, we propose a functional evaluation using graph databases such as Neo4j. Instead of providing the KG as context to an LLM, it is stored in a graph database. Then, the questions of the benchmark dataset are automatically translated into graph database queries. This evaluation has two main advantages:

1. The quality of the graph is evaluated directly.
2. The complexity of the queries can be adjusted in a controlled manner to test for reasoning over many nodes or simple knowledge retrieval.

However, this evaluation scheme is still a work in progress as it proved difficult to automatically generate accurate graph database queries suitable for reliable evaluation. This is a problem we are confident can be solved. Importantly, HyDRA-generated KGs can be loaded into Neo4j as-is (Figure 2). No structural remapping or information reduction is required, so the perceived difficulty of integrating KGs with Neo4j is limited by the generative capabilities of LLMs for query formulation, rather than graph ingestion.

6.4. Limitations and Challenges

The main limitation and challenge we encountered was the lack of benchmark datasets that are suitable for measuring the quality of a graph based knowledge representation and retrieval systems. As discussed in section 6.3 we believe that the performance numbers we present in table 6.1, while accurate for simple knowledge retrieval tasks, do not represent the full capabilities of HyDRA's ontology based approach. However, we acknowledge that compiling a benchmark with complex queries that require a well-defined and consistent graph is a difficult task. But we believe, that this is a worthwhile endeavor which will help in developing and comparing better KG construction methods, ultimately benefiting neurosymbolic applications.

7. Lessons learned

Validation We were able to identify constraints that simplified the ontology generation phase and improved output quality. In an early version of HyDRA, the ontology generation validation was less strict and allowed for multiple class hierarchies. To remediate, we implemented a contract for a subsequent ontology fixing phase to merge the class hierarchies to a single tree using a pre-defined set of operations. Later, we were able to replace this fixing phase by introducing additional (complex) constraints in the generation phase to enforce a strict hierarchy.

We assume that the constraints we identified are not exhaustive and that quality and efficiency can be improved even more by identifying further constraints.

Automatic fixing During evaluation, we observed a number of mistakes made by the LLMs which could be solved automatically without having to prompt the LLM again. In some instances, the KG generation phase yields triplets where subject and object are swapped ((`object, predicate, subject`) instead of (`subject, predicate, object`)). The resulting constraint violation lead to regeneration of the entire set of triplets output by the LLM. Instead, if such a constraint violation occurred, we now automatically swap object and subject if the resulting triplet would not violate any constraints.

We expect that there are additional opportunities to short-circuit the generation process in similar ways. By handling these cases deterministically, one could reduce uncertainty and dependence on LLM outputs.

Constraint bypasses In some instances where the generated KG triplets violates many constraints, we observed that the models simply drop all (or most of the) violating triplets instead of trying to fix them. We assume that this behavior could be reduced by either presenting errors incrementally instead of all at once, increasing both time and costs, or by requiring the model to send *delta updates* to modify the initial output accordingly.

Evaluation The observed performance gap between the ontology-enhanced and baseline approaches likely arises due to inherent biases in the evaluation setup. KG generation without an ontology effectively encodes documents directly into the context with minimal abstraction, thereby maximizing the relevance to the evaluation questions. Conversely, ontology-based KG construction does not directly depend on input documents and thus can fully represent all relevant information only if the ontology itself is infinitely detailed, which is not feasible, or if the competency questions precisely match the evaluation questions, which introduces significant evaluation bias.

Additionally, the chosen dataset primarily evaluates question-answering capabilities based on explicit information from source documents, rather than testing the type of inferential reasoning that ontologies typically support.

Therefore, future evaluations should include benchmarks specifically designed to assess the inferential and structural reasoning benefits provided by ontology-driven KG approaches. We outline one such approach in section 6.3

8. Conclusion

In this work, we introduced HyDRA, a Hybrid-Driven Reasoning Architecture that operationalizes neurosymbolic principles and Design-by-Contract concepts to guide and verify large language model (LLM)-driven knowledge graph (KG) construction. Our approach directly addresses key challenges in automated KG generation—most notably entity resolution, structural consistency, and the risk of producing isolated or semantically incoherent graph fragments—by following using verifiable contracts, a neurosymbolic approach following design-by-contract, throughout the construction pipeline.

Through experiments on the MedExQA benchmark, we observed that a baseline, ontology-free KG construction approach achieved higher raw accuracy on simple single-hop question answering tasks. While initially counterintuitive, this result reflects a limitation of current benchmarks rather than a fundamental weakness of ontology-guided methods. Our analysis shows that existing evaluation datasets fail to test the scenarios where ontology guidance offers distinct advantages: multi-hop reasoning, cross-document integration, and enforcing global structural consistency.

The HyDRA framework demonstrates that incorporating explicit constraints and iterative repair mechanisms can improve the verifiability and maintainability of automatically generated KGs, even if this comes at the cost of reduced exactness in certain narrow evaluation settings. Importantly, HyDRA opens the door for richer, contract-driven approaches where symbolic layers continuously steer and refine neural outputs.

Looking ahead, we see three key directions: (1) developing evaluation benchmarks that measure inferential reasoning and structural soundness rather than mere retrieval; (2) exploring additional constraints and deterministic fixes to further reduce LLM errors and runtime costs; and (3) extending HyDRA to support large-scale, iterative KG construction across heterogeneous domains. By advancing these lines of work, we aim to enable the reliable, scalable deployment of neurosymbolic AI systems that leverage the complementary strengths of neural generation and symbolic verification.

Declaration on Generative AI

During the preparation of this work, the authors used GPT-4 in order to: Draft content, paraphrase and reword. After using these tools the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

[1] G. F. Marcus, The Algebraic Mind: Integrating Connectionism and Cognitive Science, The MIT Press, Cambridge, MA, 2001. doi:10.7551/mitpress/1187.001.0001, bradford Books imprint.

[2] S. Auer, V. Kovtun, M. Prinz, A. Kasprzik, M. Stocker, M. E. Vidal, Towards a knowledge graph for science, in: Proceedings of the 8th International Conference on Web Intelligence, Mining and Semantics, 2018, pp. 1–6.

[3] B. C. Colelough, W. Regli, Neuro-symbolic ai in 2024: A systematic review, 2025. ArXiv preprint arXiv:2501.05435.

[4] S. Odense, A. d'Avila Garcez, A semantic framework for neuro-symbolic computing, 2024. ArXiv preprint arXiv:2212.12050.

[5] L. Zhong, J. Wu, Q. Li, H. Peng, X. Wu, A comprehensive survey on automatic knowledge graph construction, Association for Computing Machinery, 2023. Published in ACM Computing Surveys, Vol. 56, No. 4, Article 94. DOI: 10.1145/3618295.

[6] S. Carta, A. Giuliani, L. Piano, A. S. Podda, L. Pompianu, S. G. Tiddia, Iterative zero-shot llm prompting for knowledge graph construction, 2023. ArXiv preprint arXiv:2307.01128.

[7] B. Zhang, H. Soh, Extract, define, canonicalize: An llm-based framework for knowledge graph construction, 2024. ArXiv preprint arXiv:2404.03868.

[8] T. R. Besold, A. d'Avila Garcez, S. Bader, H. Bowman, P. Domingos, P. Hitzler, K.-U. Kuehnberger, L. C. Lamb, D. Lowd, P. Machado Vieira Lima, L. de Penning, G. Pinkas, H. Poon, G. Zaverucha, Neural-symbolic learning and reasoning: A survey and interpretation, 2017. ArXiv preprint arXiv:1711.03902.

[9] H. Paulheim, Knowledge graph refinement: A survey of approaches and evaluation methods, SAGE Publications, 2016, pp. 489–508. Published in the journal Semantic Web, Vol. 8, No. 3.

[10] S. Li, Z. Wang, X. Zhang, et al., Knowledge graphs, large language models, and hallucinations: An nlp perspective, arXiv preprint arXiv:2411.14258 (2024).

[11] A. Bayer, H. Paulheim, Assessing llms suitability for knowledge graph completion, arXiv preprint arXiv:2405.17249 (2024).

[12] P. A. Bonatti, S. Decker, A. Polleres, V. Presutti, Knowledge graphs: New directions for knowledge representation on the semantic web (dagstuhl seminar 18371), Dagstuhl Reports 8 (2019) 29–111. doi:10.4230/DagRep.8.9.29.

[13] H. B. Newcombe, J. M. Kennedy, S. J. Axford, A. P. James, Automatic linkage of vital records, Science 130 (1959) 954–959. doi:10.1126/science.130.3381.954.

[14] M.-C. Dinu, C. Leoveanu-Condrei, W. Zellinger, M. Holzleitner, S. Hochreiter, Symbolicai: A framework for logic-based approaches combining generative models and solvers, in: Proceedings of the 3rd Conference on Lifelong Learning Agents (CoLLAs), 2024. ArXiv:2402.00854.

[15] F. Al Machot, M. T. Horsch, H. Ullah, Designing the conceptual landscape for a xair validation infrastructure, in: Building Trustworthy AI: Transparent AI Systems via Language Models, Ontologies, and Logical Reasoning (TranspNet), volume 1375 of *Lecture Notes in Networks and Systems*, Springer Nature, 2024, pp. 25–34. First Online: 16 May 2025.

[16] B. Meyer, Object-Oriented Software Construction, 2nd ed., Prentice-Hall, Upper Saddle River, NJ, USA, 1997.

[17] A. Gangemi, V. Presutti, D. R. Recupero, A. G. Nuzzolese, F. Draicchio, M. Mongiovì, FRED: Frame-Based Event Recognition and Description, Semantic Web 8 (2017) 821–832.

[18] P. Robinson, S. Singh, J. Morris, GraphRAG: Augmenting LLMS with Knowledge Graphs, 2023. arXiv:2310.XXXX.

[19] R. Zhang, X. Chen, L. Gui, Y. Chen, Y. Tang, Y. Zhang, AdaKGC: Schema-Adaptable KG Construction, in: Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 2023, pp. 1–15.

[20] H. Knublauch, D. Kontokostas, Shapes Constraint Language (SHACL), W3C Recommendation, 2017. URL: https://www.w3.org/TR/shacl/.

[21] A. Funk, T. Kliegr, T. Soru, A. Primpeli, T. Pellegrini, A. Angioni, LLMs4OL: A Framework for Large-Scale Ontology Learning, in: Proceedings of the ISWC 2023 Posters, Demos and Industry Tracks: From Novel Ideas to Industrial Applications co-located with 22nd International Semantic Web Conference (ISWC 2023), 2023.

[22] T. Baldwin, J. H. Lau, K. Verspoor, OLLM: End-to-End Ontology Construction with Regularised LLMs, in: Findings of the Association for Computational Linguistics: ACL 2024, 2024.

[23] M. Gruninger, M. S. Fox, The role of competency questions in enterprise engineering, in: Proceedings of the IFIP WG5.7 Workshop on Benchmarking - Theory and Practice, 1995, pp. 22–29.

[24] Z. Pan, K. Zhao, Y. Luo, Retrieval-Augmented CQ Generation for Scientific Domains, in: Proceedings of the Extended Semantic Web Conference (ESWC), 2024.

[25] H. Wu, Y. Chen, Y. Wang, J. Li, Multi-persona prompting for diverse cqs, 2024. arXiv:2404.XXXX.

[26] J. Brank, M. Grobelnik, D. Mladenić, A survey of ontology evaluation techniques, Journal on Data Semantics IV (2005) 132–157.

[27] C. M. Keet, What We Still Don't Evaluate in KGs, Semantic Web Journal 13 (2022) 1237–1250.

[28] L. Ribeiro, G. Costa, G. de Melo, KGrEaT: KG Quality via Down-stream Tasks, in: Proceedings of the 30th ACM International Conference on Information & Knowledge Management (CIKM '21), 2021, pp. 1569–1578.

[29] D. Daza, M. Cochez, G. de Melo, GraphRAFT: Fine-Tuning LLMs for Cypher Query Writing, in: Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL), 2024.

[30] B. Meyer, Applying "design by contract", IEEE Computer 25 (1992) 40–51. doi:10.1109/2.161279.

[31] S. Mohajerani, I. Hasuo, K. Komorida, Y. Sato, A theory of probabilistic contracts, in: Proceedings of the 26th International Symposium on Formal Methods (FM), 2024. Forthcoming.

[32] X. Bian, A. Tor-Shell, P. Tabuada, Pacti: Assume-Guarantee Contracts at Scale, in: Proceedings of the International Conference on Embedded Software (EMSOFT), 2023.

[33] A. Ghosh, R. Li, Z. Zhang, O. Bastani, ML-Guard: Runtime Contract Enforcement for ML Pipelines, in: Proceedings of the 2023 USENIX Annual Technical Conference (USENIX ATC '23), 2023.

[34] NVIDIA, NeMo Guardrails, https://github.com/NVIDIA/NeMo-Guardrails, 2023. Accessed: 2025-06-11.

[35] C. A. R. Hoare, An axiomatic basis for computer programming, Communications of the ACM 12 (1969) 576–580. doi:10.1145/363235.363259.

[36] L. Lamport, Proving the correctness of multiprocess programs, IEEE Transactions on Software Engineering SE-3 (1977) 125–143. doi:10.1109/TSE.1977.229904.

[37] Y. Kim, J. Wu, Y. Abdulle, H. Wu, Medexqa: Medical question answering benchmark with multiple explanations, 2024. arXiv:2406.06331.

[38] OpenAI, GPT-4.1, Large language model, 2025. URL: https://platform.openai.com/, released April 14, 2025; accessed via OpenAI API.