Formalization of Deductive Rules for Syntactic-Semantic Transformations in a System for Ontology Construction from Textual Documents

Andrii Chornyi
dept. Information Systems and Networks
Lviv Polytechnic National University
Lviv, Ukraine

Dmytro Dosyn
dept. Information Systems and Networks
Lviv Polytechnic National University
Lviv, Ukraine

***Abstract* — This paper presents a novel framework for formalizing and storing deductive rules that govern syntactic-semantic transformations in automated ontology construction from textual data. As unstructured text continues to dominate modern information landscapes, converting it into structured, machine-understandable knowledge is critical for advancing intelligent systems. Leveraging a rule-based approach rooted in formal linguistic and logical representations, we highlight its strengths in precision, interpretability, and adaptability for ontology extraction. We introduce a graph-based abstraction for representing transformation rules and storing them in graph databases, which enables efficient pattern matching, dynamic rule management, and introspective analysis for continuous optimization. By conceptualizing these transformations as ETL (Extract, Transform, Load) processes, our method provides a clear framework for evaluating and refining rule sets. The approach is implemented on the CROCUS platform, integrating Stanford CoreNLP for linguistic parsing with Apache Jena Fuseki for semantic storage. Experimental results demonstrate the practical effectiveness of our method in enhancing the development of intelligent agents through robust and explainable rule formalization.**

Keywords — ontology construction, deductive rules, syntactic-semantic transformation, graph database, rule-based approach, natural language processing, knowledge extraction, ETL process, reflective analysis

I. Introduction

In the context of the modern information society, the majority of knowledge is represented in electronic documents as unstructured textual data. The effective transformation of such data into a structured form is a critical prerequisite for the advancement of numerous fields, including science, business, and information technology. Natural Language Processing (NLP) methods serve as key tools in this process, providing means for the automated analysis, interpretation, and extraction of meaningful information from texts. This, in turn, lays the foundation for the development of intelligent agents capable of autonomous learning across various subject domains — from broad to highly specialized areas.

One of the approaches to extracting structured data from text is the rule-based approach, which involves the use of predefined linguistic or logical deductive rules to identify and interpret relevant information in unstructured textual sources. This approach is grounded in the formalization of knowledge about linguistic constructions, patterns, and contexts that describe target entities, their properties, or relationships.

In order to ensure the reproducibility, scalability, and flexibility of syntactic-semantic transformation processes, there arises a need to develop a formalized abstraction for representing deductive rules—one that supports their efficient storage, analysis, and execution. The analysis of rules plays a particularly important role, as it enables clustering and optimization of their execution based on statistics from previous applications, thereby enhancing the overall efficiency of the system. This approach represents a form of first-order reflection, allowing not only for the reduction of redundancy within the rule set, but also for the identification of usage patterns—an essential prerequisite for the adaptive configuration of deduction mechanisms in accordance with the task context and the dynamics of incoming data.

II. A Review of Existing Approaches to Ontology Construction from Text

In the domain of automated ontology learning, the deductive, or rule-based, approach relies on the use of formalized logical rules to infer new knowledge from existing assertions. This method offers several advantages, particularly in contexts where accuracy, controllability, and explainability of results are of critical importance.

Transparency and Explainability. Deductive systems enable clear traceability of the knowledge inference process, which is critically important in domains where trust in the results is essential, such as medicine, law, and education. The use of formalized rules provides a comprehensible pathway from input data to derived conclusions [1].

Controllability and Precision. Since the rules are defined by experts or formalized based on domain-specific knowledge, deductive systems exhibit high precision in knowledge inference. This is particularly important in contexts where errors can have serious consequences, such as in decision support systems [2].

Adaptability and Scalability. Deductive systems can be adapted to various subject domains through the modification or extension of the rule set. This enables efficient scaling of knowledge systems without the need for complete retraining of the model [3].

Efficiency in Combination with Graph Databases. The integration of deductive methods with graph databases allows for the efficient storage and processing of complex ontological structures, providing rapid access to knowledge and enabling its updating [4].

Capacity for Reflective Analysis. Deductive systems can be enhanced with mechanisms that allow for the analysis and optimization of the rules themselves based on the statistics of their application. This opens up possibilities for automatic clustering, detection of redundant or conflicting rules, and their adaptation to new contexts [5].

The rule-based method for ontology construction from text was proposed in the OntoHarvester system [6], which introduced a novel approach to the automatic creation of domain-specific ontologies from a small corpus of texts using deep NLP analysis. OntoHarvester starts with a small set of concepts (seeds) and iteratively expands the ontology by adding new terms that have strong semantic connections with existing concepts. According to the authors, this approach allows for the creation of comprehensive ontologies from small text corpora while remaining resilient to noise and focused on a specific domain (subject area).

According to the findings of their research, the authors [6] claim that their proposed approach achieves higher accuracy and coverage compared to other methods, while also demonstrating resilience to noise in the text and the ability to create comprehensive ontologies from small text corpora. This result is achieved, in particular, through the use of GD-rules (Graph Domain Rules), which are templates used for analyzing text graphs (graphs that represent grammatical relationships between terms in the text). These rules, with a syntax similar to SPARQL, allow for the definition of complex patterns for searching relationships within the graphs.

According to [6], GD-rules are more powerful compared to traditional approaches that use tree-like templates or regular expressions. They allow for the consideration of complex grammatical structures in the text, which significantly improves the accuracy of extracting ontological relationships. Furthermore, unlike statistical methods, GD-rules do not require large volumes of data for training, making them more practical for working with small text corpora.

A similar deductive approach, based on linguistic patterns, is discussed in the article [7] for the extraction of factual information. The authors present some basic correspondences between linguistic patterns and their ontological interpretations in canonical form.

Another practical idea and its implementation are presented in the article [8]. The authors propose the development of a question-answering (QA) system based on the transformation of queries formulated in natural language into SPARQL queries, which are used in semantic web applications to retrieve data from graph databases. A vast number of such resources are interconnected through the Linked Open Data Cloud (Linked Open Data Cloud, n.d.), providing users with direct access to thousands of RDF/RDFS datasets via SPARQL endpoints.

The primary issue addressed in this work is the complexity of using the SPARQL query language for non-expert users. The proposed methodology involves transforming an unstructured natural language question into structured constructs that can be processed by a machine – similarly to previous works – through a rule-based approach.

The reviewed studies clearly demonstrate the rationale for using a graph database both as a knowledge storage environment and as an operational medium for performing transformations between input data – primarily derived from text parsers – and output data, such as knowledge graphs or ontologies. However, the question of how to store syntactic-semantic transformation rules remains open.

The use of a graph database for storing the formal representation of graph transformation rules is well-justified, particularly in the context of transforming syntactic graphs – such as parse trees or dependency graphs – into semantic graphs, such as RDF triples or conceptual graphs. This rationale is grounded in the following conceptual considerations.

Graph Structure of Rules. Transformation rules inherently possess a graph-based nature. Each rule typically consists of a left-hand side (LHS), which defines the structure to be identified in the syntactic graph, and a right-hand side (RHS), which specifies the structure to be generated in the semantic graph. Since both the input and output of a rule are graphs, it is reasonable to represent the rules themselves as graph structures. Graph databases are well-suited for storing such representations [10].

Efficiency of Pattern Matching. Graph transformations heavily rely on subgraph matching within larger graphs. Graph databases are optimized for such pattern-matching and graph traversal operations, making them ideal for the efficient execution and management of transformation rules [11].

Modeling Interrelationships Between Rules. In many systems, transformation rules are not isolated; they may exhibit dependencies (e.g., one rule must be applied before another), belong to specific categories, or possess hierarchical relationships. Graph databases enable the efficient modeling and querying of such interrelations among rules through edges connecting the nodes that represent them [12] [13] [14].

Storage of Metadata and Semantics. Graph databases support the attachment of properties (attributes) to nodes and edges, which facilitates the storage of additional information about each rule – such as its priority, application conditions, or links to ontological concepts. This capability enhances the management of semantic richness within the transformation system [15].

Tracking the Evolution and Provenance of Rules. In real-world applications, rules evolve over time. There may arise a need for versioning, tracking changes, or recording the moment and way they are applied. A graph database enables the implementation of provenance tracking mechanisms by storing the history of changes as part of the graph structure [16].

III. Objective of the Study

The objective of this study is to develop an abstraction for representing and storing syntactic-semantic transformation rules in a graph database, with the capability for their further analysis. In the long term, the results of this work are intended to serve as the foundation for implementing reflection methods in an intelligent agent and pave the way for realizing an effective self-learning process.

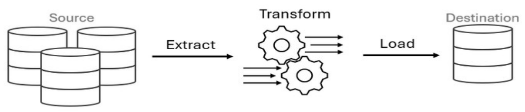

Fig. 1. Typical ETL Process Diagram.

IV. Evaluation of the Complexity of Deductive Syntactic-Semantic Transformation Rules

To implement rule-oriented transformations, an effective tool for storing and processing operational syntactic and semantic data is required. Given the nature of this data, the use of a graph database is appropriate, where elements such as words, entities, annotations, etc., are represented as nodes of the graph, and the relationships between them are represented as edges. This approach allows for a more natural representation of nonlinear relationships (e.g., dependencies between words in different parts of a sentence or coreference between entities).

The main operations when working with a graph are:

- Adding a node

$$V_i = \{A_1, A_2, \dots, A_k\},$$

where $A_1, A_2, \dots, A_k$ are the attributes of the node (e.g., token, POS tag), which can also be nodes themselves.

- Adding an edge (link)

$$E_{ij} = (V_i, V_j, R),$$

where V_i, V_j are nodes, and R is the type of relationship (e.g., syntactic dependency).

- Searching for a link

$$T(V_i, V_j) = \min_P \sum_{(V_k, V_m) \in P} \omega_{km},$$

where P is the path between V_i, V_j, and ω_{km} is the weight of the link.

The complexity of the approach using a graph is as follows:

- Adding a node or edge: $O(1)$
- Searching for neighboring nodes: $O(1)$ (average for a well-structured graph).
- Searching for relationships between entities: $O(\log n)$ or $O(d)$,

where n is the total number of nodes (vertices) in the graph, d is the average node degree:

$$d = \frac{2E}{V},$$

where V is the total number of nodes (vertices) in the graph,
E is the total number of edges (connections) in the graph.

Thus, the overall complexity of this approach will be:

$$MAX(O(\log n), O(1))$$

Furthermore, the graph database allows for easy formulation, storage, and application of rules in the rule-based approach to ontology construction from text, where a rule can be represented as follows:

$$q_{sel}(G_{syn}) \rightarrow q_{ins}(G_{sem}),$$

where q_{sel}, q_{ins} are queries (select, insert, respectively) to the graph database, G_{syn} is the graph of syntactic relations, and G_{sem} is the graph of semantic relations.

V. Formalization of Deductive Rules for Syntactic-Semantic Transformations

In general, the set of rules represents a collection $Q = \{Q_1, Q_2, \dots, Q_k\}$. When applying syntactic-semantic transformations over graphs, the rules are represented as textual data in a well-known format, such as SPARQL or Cypher. This allows them to be easily stored, transmitted, and applied on the fly. Each rule from the specified set takes the form:

$$Q_i = (q_{sel\,i}, q_{ins\,i}),$$

where $q_{sel\,i}$ represents the previously mentioned left-hand side (LHS), and $q_{ins\,i}$ represents the right-hand side (RHS).

The presented rules indicate that the transformation stage is blurred between $q_{sel\,i}$ and $q_{ins\,i}$, and may be a component of both the left-hand side (LHS) and the right-hand side (RHS). This representation is comprehensive for mapping rules, but insufficient for transformation rules. The latter also include a transformation function $T(q_{sel\,i})$:

$$Q_i = (q_{sel\,i}, T(q_{sel\,i}), q_{ins\,i}).$$

Thus, from the perspective of Data Engineering, the rule-based approach to ontology construction from text represents a typical ETL (Extract, Transform, Load) process, as illustrated in Fig. 1.

The formalization of syntactic-semantic rules in the form of ETL processes opens new horizons in the research and development of intelligent agents capable of learning from unstructured textual documents.

The equations are an exception to the prescribed specifications of this template. You will need to determine whether or not your equation should be typed using either the Times New Roman or the Symbol font (please no other font). To create multileveled equations, it may be necessary to treat the equation as a graphic and insert it into the text after your paper is styled. In particular, the separation of ETL stages enables the evaluation of each individual rule, which opens up opportunities for analyzing their effectiveness without increasing the complexity of the core algorithm, keeping it at the level of $MAX(O(\log n), O(1))$.

```
PREFIX crocus: http://crocus.science/
DROP GRAPH <http://localhost:3330/opgraph/statrule>;
INSERT
{
//----- LOAD -----
  GRAPH <http://localhost:3330/opgraph/semgraph>
  {
    ?fulln ?firstName ?fn .
    ?fulln ?lastName ?ln .
    ?fulln ?rdfType ?rdfPerson .
  }
//----- Reflection -----
  GRAPH <http://localhost:3330/opgraph/statout>
  {
    ?fulln ?firstName ?fn .
    ?fulln ?lastName ?ln .
    ?fulln ?rdfType ?rdfPerson .
  }
  GRAPH <http://localhost:3330/opgraph/statin>
  {
    ?subject ?compound       ?object .
    ?subject ?namedEntityTag ?parserTypePerson .
    ?object  ?namedEntityTag ?parserTypePerson .
  }
  GRAPH <http://localhost:3330/opgraph/statrule>
  {
    ?ruleId ?ruleStart ?startTime .
  }
}
WHERE
{
  GRAPH <http://localhost:3330/opgraph/syngraph>
  {
    BIND(NOW() AS ?startTime)
//----- EXTRACT -----
    BIND(<http://nlp.stanford.edu#compound> AS ?compound)
    BIND(<http://nlp.stanford.edu#NamedEntityTagAnnotation>
         AS ?namedEntityTag)
    BIND("PERSON" AS ?parserTypePerson)

    ?subject ?compound       ?o .  BIND(IRI(?o) AS ?object)
    ?subject ?namedEntityTag ?parserTypePerson .
    ?object  ?namedEntityTag ?parserTypePerson .

//----- TRANSFORM -----
    BIND(IRI(crocus:PersonFirstLastNameRule) AS ?ruleId)
    BIND(IRI(crocus:startTime) AS ?ruleStart)
    BIND(<rdf:typeof> AS ?rdfType)
    BIND("person" AS ?rdfPerson)
    BIND(IRI(CONCAT("crocus:",
REPLACE(STR(?object), ".*[/#]", ""),
      "_",
REPLACE(STR(?subject), ".*[/#]", "")))
AS ?fulln)
    BIND(<crocus:firstname> AS ?firstName)
    BIND(<crocus:lastname> AS ?lastName)
    BIND(?object  AS ?fn)
    BIND(?subject AS ?ln)
  }
};
//----- Reflection -----
INSERT
{
  GRAPH <http://localhost:3330/opgraph/statrule>
  {
    ?ruleId ?ruleEnd ?endTime .
    ?ruleId ?ruleDuration ?duration .
  }
}
WHERE
{
  GRAPH <http://localhost:3330/opgraph/statrule>
  {
    ?rule ?ruleStart ?startTime .
  }
  BIND(IRI(crocus:PersonFirstLastNameRule) AS ?ruleId)
  FILTER(?rule = ?ruleId)
  FILTER(?ruleStart = IRI(crocus:startTime))
  BIND(IRI(crocus:endTime) AS ?ruleEnd)
  BIND(IRI(crocus:duration) AS ?ruleDuration)
  BIND(NOW() AS ?endTime)
  BIND((?endTime - ?startTime) AS ?duration)
};
```

From an epistemological standpoint, this approach constitutes an implementation of first-level reflection in an intelligent agent, which in this case is manifested in the agent's ability to evaluate its own activity – specifically, the process of ontology construction from text.

To demonstrate the validity and effectiveness of this approach, a study was conducted on a syntactic-semantic rule with the following logic:

- if there exists a "compound" relation between two nodes in the syntactic graph, and both the subject and object of this relation are, in turn, connected to the literal "PERSON" via a *NamedEntityTagAnnotation* link, then a node of type "person" should be created in the semantic graph, with the subject of the "compound" relation as the first name and the object as the last name.

The implementation of this rule was carried out at the level of the graph database using the SPARQL query provided below. The relative simplicity of the rule made it possible to do so without separating the ETL processes at the application level. This implementation enabled the analysis to be conducted without introducing technological delays required for executing Java code. The study was conducted using the previously developed CROCUS platform, which utilizes Stanford CoreNLP as the text parser and Apache Jena Fuseki as the embedded graph database server.

In the provided SPARQL code, the processes of extracting, transforming, and loading (ETL) are highlighted through comments, as well as the reflection blocks.

In this SPARQL script, the input and output data of the rule are additionally recorded in the *statin* and *statout* graphs (representing the Extract and Load stages, respectively), as well as the execution duration of the rule in the *statrule* graph (representing the Transform stage).

The study started with a text corpus of 20 sentences, adding one sentence at a time from 1 to 20. However, this volume of text did not cause significant changes in the rule's execution time that were sufficient for analysis. Therefore, a nonlinear increment was subsequently applied, doubling the number of sentences in the text corpus at each step up to 640 sentences. To provide a better understanding of the actual volume of the text corpus, it should be noted that 80 sentences correspond to approximately one A4 page, formatted in Times New Roman font size 12 with 1 cm margins on all sides.

The text data were generated using an LLM system with the following requirements: each sentence must mention a single person in the format of a <First Name, Last Name> combination, with each combination being unique.

$E_{EXTRACT}$first stage of the experiment, a clear dependency was observed between the execution time of the syntactic-semantic transformation and, to a great extent, the number of connections (edges) in the $E_{EXTRACT}$ sample during the Extract stage (pattern search in the syntactic graph) (TABLE I). The correlation coefficient between these indicators is 0.98, which significantly outweighs the correlation coefficient between execution time and the number of edges in the syntactic graph during the incremental increase in the text volume, when adding extra sentences at each step.

TABLE I. THE DEPENDENCY OF RULE EXECUTION TIME ON THE NUMBER OF EDGES IN THE SAMPLE .

N_{sent}	T_{full}	$E_{EXTRACT}$
1	0.022	3
2	0.027	6
3	0.021	9
4	0.020	12
5	0.029	15
6	0.028	18
7	0.038	21
8	0.031	24
9	0.041	27
10	0.028	30
11	0.029	33
12	0.042	36
13	0.038	39
14	0.031	42
15	0.039	45
16	0.042	48
17	0.047	51
18	0.048	54
19	0.037	57
20	0.043	60
40	0.049	116
80	0.059	176
160	0.078	255
320	0.100	384
640	0.133	567
		0.98

N_{sent} – the number of sentences in the text corpus;

T_{full} – the execution time of the rule (script) in the full text corpus addition mode;

$E_{EXTRACT}$ – the number of connections (edges) extracted from the syntactic graph during the Extract stage;

$r(T_{full})$ – the Pearson linear correlation between T_{full} and the parameter $E_{EXTRACT}$.

This pattern indicates the presence of a caching function in the graph database server used in the study (Apache Jena Fuseki), as the syntactic analysis data from the previous step were not removed during the incremental increase in the text corpus size. Therefore, the data from the Extract stage of the rule being studied at each step (except the first) are partially cached.

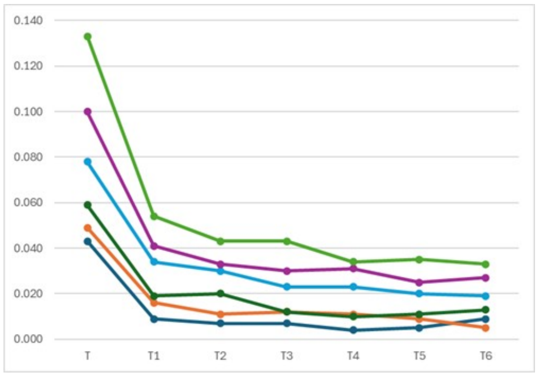

Fig. 2

To further investigate the impact of data caching in the graph database server on the execution time of syntactic-semantic rules, an additional study was conducted. This study focused on analyzing the change in execution time during repeated rule executions. The results of this study are presented in TABLE II. The research algorithm involved executing the rule ten times, measuring the time after its initial execution on the freshly loaded syntactic graph for various text corpus sizes.

TABLE II. THE IMPACT OF CACHING ON THE REPEATED EXECUTION OF THE SYNTACTIC-SEMANTIC RULE.

N_{sent}	T_{full}	T_1	T_2	T_3	T_4	T_5	T_6
20	0.043	0.009	0.007	0.007	0.004	0.005	0.009
40	0.049	0.016	0.011	0.012	0.011	0.009	0.005
80	0.059	0.019	0.020	0.012	0.010	0.011	0.013
160	0.078	0.034	0.030	0.023	0.023	0.020	0.019
320	0.100	0.041	0.033	0.030	0.031	0.025	0.027
640	0.133	0.054	0.043	0.043	0.034	0.035	0.033

N_{sent} - the number of sentences in the text corpus;

T_{full} - the execution time of the rule after adding the text corpus and its initial execution;

$T_1 \dots T_6$ - the execution time of the rule during each subsequent attempt of repetition.

As shown in TABLE II, each subsequent execution of the rule becomes faster until a certain minimum is reached, which further confirms the presence of a query caching function that helps optimize rule-based methods for ontology construction from text. For clarity, a chart illustrating the change in rule execution time as a function of the number of repetitions is provided on .

The results of this study clearly demonstrate that data caching occurs during the Extract stage (Fig. 1), and each subsequent execution of the rule is significantly faster. Therefore, to optimize the performance of rule-based approaches for ontology construction from text, it makes sense to group rules based on the similarity of input data (samples during the Extract stage). This approach greatly accelerates the execution of syntactic-semantic transformations, as running rules in groups ensure that each subsequent rule benefits from a partially or fully cached set of input data.

This highlights the importance of formalizing rules into a graph representation. By representing rules in a structured, graph-based format, it becomes easier to identify patterns, group similar rules together, and leverage caching mechanisms more effectively. Graph representation allows for a more efficient mapping of relationships and enables optimization strategies like rule grouping, which can significantly reduce execution times and improve scalability.

It is evident that data caching occurs at the Extract stage (Fig. 1), and each subsequent execution of a rule is significantly faster. Therefore, to optimize the performance of rule-based approaches for ontology construction from text, it is reasonable to group rules based on the similarity of their input data (i.e., the samples obtained at the Extract stage). This approach considerably accelerates the execution of syntactic-semantic transformations, as executing rules in groups allows each subsequent rule to operate on a partially or even fully cached set of input data.

This underscores the importance of formalizing rules in the form of a graph-based representation. By representing rules in a structured, graph-oriented format, it becomes easier to identify patterns, group similar rules, and more effectively leverage caching mechanisms. A graph-based representation enables more efficient modeling of interdependencies and facilitates the implementation of optimization strategies, such as rule grouping, which can significantly reduce execution time and enhance scalability.

```
@prefix crocus: <http://crocus.science/> .
@prefix nlp:    <http://nlp.stanford.edu#> .
@prefix  rdf:      <http://www.w3.org/1999/02/22-rdf-syntax-
ns#> .
@prefix ex:     <http://example.org#> .

### --- EXTRACT ---
ex:subject nlp:compound ex:object .
ex:subject nlp:NamedEntityTagAnnotation "PERSON" .
ex:object  nlp:NamedEntityTagAnnotation "PERSON" .

### --- TRANSFORM ---
ex:subject nlp:compound ex:object .
ex:subject nlp:NamedEntityTagAnnotation "PERSON" .
ex:object  nlp:NamedEntityTagAnnotation "PERSON" .

[] rdf:type crocus:GeneratedNode ;
           crocus:constructedFrom [
               crocus:part ex:object ;
               crocus:part ex:subject ;
               crocus:pattern "crocus:{object}_{subject}"
           ] .

### --- LAOD ---
crocus:object_subject crocus:firstname ex:object .
crocus:object_subject crocus:lastname  ex:subject .
crocus:object_subject rdf:type         "person"  .
```

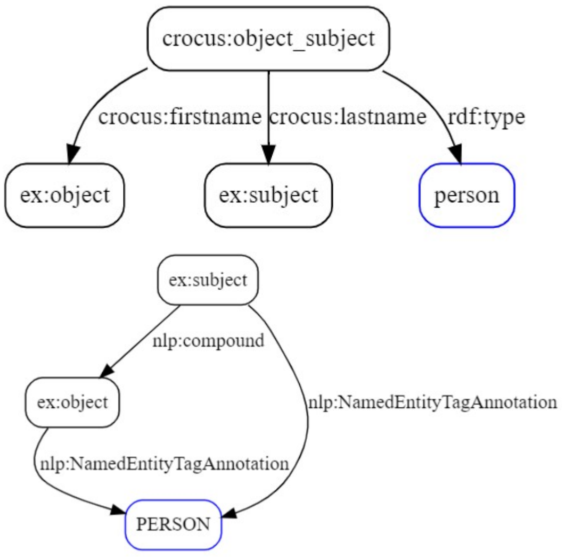

Fig. 2. Visualization of the graph representation of the LOAD stage of the syntactic-semantic transformation.

Having structured the syntactic-semantic rule in the form of an ETL process (Fig. 1), the next step is to formalize each stage as a graph-based representation. Accordingly, using the previously examined rule, a prototype of its graph representation was developed in RDF format.

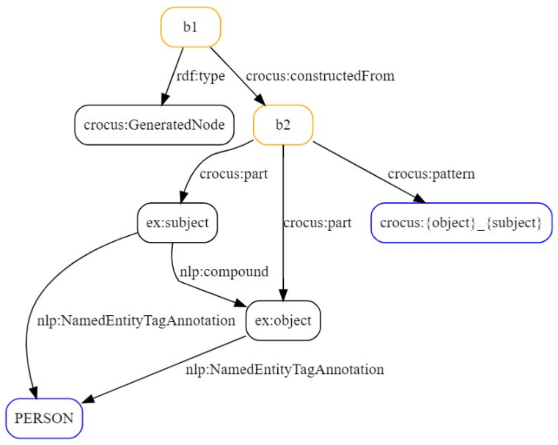

Fig. 3. Visualization of the graph representation of the EXTRACT stage of the syntactic-semantic transformation.

To illustrate the structure of the ETL-stage graphs involved in the syntactic-semantic transformation of the examined rule, visualizations were created (Fig. 4, Fig. 3, Fig. 2) using the online tool available at https://semantechs.co.uk/turtle-editor-viewer/.

VI. CONCLUSIONS

This study introduces a formalized and scalable framework for representing syntactic-semantic transformation rules as graph structures within ontology construction systems. By conceptualizing rule execution as an ETL process and embedding it in a graph database environment, we enable efficient rule management, reflective analysis, and dynamic optimization. Experimental validation on the CROCUS platform confirms the practical viability of this approach: rules expressed in SPARQL can be effectively executed, monitored, and improved in real-time. Notably, we demonstrate that graph-based formalization not only facilitates caching and performance optimization but also opens the door to rule clustering and dependency modeling—critical capabilities for adaptive, intelligent agents. These findings pave the way for next-generation knowledge extraction systems that are both transparent and adaptable, capable of learning from and evolving with incoming unstructured text data. Future work will explore extending this reflective rule framework to support autonomous rule evolution and cross-domain knowledge transfer in semantically rich environments.

REFERENCES

[1] K. Baclawski, M. Bennett, . G. Berg-Cross, T. Schneider, R. Sharma, M. Underwood та A. Westerinen, «Toward Trustworthy AI Systems,» Washington Academy of Sciences, p. 110, 2024.

[2] Z. Jin, L. Jin, Y. Luo, S. Feng, Y. Shi та K. Zheng, «Correctness Learning: Deductive Verification Guided Learning for Human-AI Collaboration,» (Preprint) researchgate.net, 10 Mar 2025.

[3] J. Rane, S. K. Mallick, Ö. Kaya та N. L. Rane, «Scalable and adaptive deep learning algorithms for large-scale machine learning systems,» Deep Science, pp. 39-92, October 2024.

[4] M. . Á. Rodríguez-García та R. Hoehndorf, «Inferring ontology graph structures using OWL reasoning,» BMC Bioinformatics, 2018.

[5] S. Hassanpour, M. J. O'Connor та A. K. Das, «Clustering Rule Bases Using Ontology-Based Similarity Measures,» Journal of Web Semantics, pp. 1-8, 2014.

[6] H. Mousavi, D. Kerr, M. Iseli та C. Zaniolo, «Harvesting Domain Specific Ontologies from Text,» в International Conference on Semantic Computing, Newport Beach, CA, USA, 2014.

[7] A. Doroshenko, «Development of information technology for intellectual analysis of factographic information,» Bionics of Intelligence, т. 1 (90), pp. 116-121, 2018.

[8] N. Zlatareva та D. Amin, «Processing Natural Language Queries in Semantic Web Applications,» в The 7th World Congress on Electrical Engineering and Computer Systems and Science, 2021.

[9] «Linked Open Data Cloud,» [Онлайновий]. Available: https://www.lod-cloud.net/.

[10] M. N. Roelofs та R. Vos, «Automatically inferring technology compatibility with an ontology and graph rewriting rules,» Journal of Engineering Design, № 32 (2), pp. 90-114, 2020.

[11] M. Fuchs, The Matching-Graph, 2023.

[12] L. C. Shimomura, «Graph Profiling with Graph Generating Dependencies,» в CEUR Workshop Proceedings, 2022.

[13] W. Fan та C. Hu, «Big Graph Analyses: From Queries to Dependencies and Association Rules,» Springer Nature, pp. 36-55, 2017.

[14] D. Liu, S. Kwashie, Y. Zhang, G. Zhou, M. Bewong, X. Wu, X. Guo, K. He та Z. Feng, «An Efficient Approach for Discovering Graph Entity Dependencies,» SSRN, 21 Sep 2023.

[15] I. Robinson, J. Webber та E. Eifrem, Graph Databases, 2nd Edition, O'Reilly Media, Inc., 2015.

[16] H. Dibowski, «Full Traceability and Provenance for Knowledge Graphs,» Formal Ontology in Information Systems, pp. 223-237, December 2024.