AgentODRL: A Large Language Model-based Multi-agent System for ODRL Generation

Wanle Zhong, Keman Huang*, Xiaoyong Du

School of Information, Renmin University of China, Beijing, China
{wanle, keman, duyong}@ruc.edu.cn

Abstract

The Open Digital Rights Language (ODRL) is a pivotal standard for automating data rights management. However, the inherent logical complexity of authorization policies, combined with the scarcity of high-quality "Natural Language-to-ODRL" training datasets, impedes the ability of current methods to efficiently and accurately translate complex rules from natural language into the ODRL format. To address this challenge, this research leverages the potent comprehension and generation capabilities of Large Language Models (LLMs) to achieve both automation and high fidelity in this translation process. We introduce AgentODRL, a multi-agent system based on an Orchestrator-Workers architecture. The architecture consists of specialized Workers, including a Generator for ODRL policy creation, a Decomposer for breaking down complex use cases, and a Rewriter for simplifying nested logical relationships. The Orchestrator agent dynamically coordinates these Workers, assembling an optimal pathway based on the complexity of the input use case. Specifically, we enhance the ODRL Generator by incorporating a validator-based syntax strategy and a semantic reflection mechanism powered by a LoRA-finetuned model, significantly elevating the quality of the generated policies. Extensive experiments were conducted on a newly constructed dataset comprising 770 use cases of varying complexity, all situated within the context of data spaces. The results, evaluated using ODRL syntax and semantic scores, demonstrate that our proposed Orchestrator-Workers system, enhanced with these strategies, achieves superior performance on the ODRL generation task.

Code — https://github.com/RUC-MAS/AgentODRL

1 Introduction

With the burgeoning development of the digital economy, the importance of Data Spaces as a distributed infrastructure to facilitate data exchange among multiple organizations, while ensuring the principles of data sovereignty and interoperability (Otto and Jarke 2019), has become increasingly prominent. In such ecosystems, to achieve trusted cross-entity data sharing and circulation (Nagel et al. 2023; Farrell et al. 2023), initiatives like the International Data Spaces Association (IDSA) have adopted the W3C's Open Digital Rights Language (ODRL) as the core standard for describing usage policies for data assets (Dam, Krimbacher, and Neumaier 2023; Meckler et al. 2023; Villata and Iannella 2018; Akaichi et al. 2024). However, the creation of ODRL policies is deeply dependent on Semantic Web technologies, requiring policy authors to be familiar with the RDF graph data model, its serialization formats (Kellogg, Champin, and Longley 2019), and the complex conceptual system of ODRL itself. This presents a significant barrier to adoption for domain experts without a technical background. Even today, with Large Language Models (LLMs) demonstrating powerful language understanding and generation capabilities, bridging this technical divide to accurately and reliably translate complex natural language (NL) rules from legal regulations and commercial agreements into structurally rigorous ODRL policies remains a core bottleneck.

Traditionally, methods for handling such translation tasks have tended to adopt a monolithic architecture, relying on a single LLM to perform the entire end-to-end generation process from NL to ODRL. Such approaches, from the earlier "Ontology-Guided" generation (Kotis, Vouros, and Spiliotopoulos 2020) to subsequent methods that introduce "Self-Correction Rules" (SCR) for optimization (Mustafa et al. 2025), essentially attempt to address all types of complexity with a single, unified model. However, this architecture often proves inadequate when dealing with multiple, cognitively distinct sub-tasks concurrently—such as parsing dependencies in legal text, performing semantic segmentation, and executing constrained generation within a strict syntax. Its fundamental flaw lies in the inability of a single model to concurrently handle these diverse cognitive challenges. Consequently, its performance degrades significantly, especially when faced with policies containing complex logical structures like parallel or recursive dependencies, failing to guarantee the accuracy and completeness of the generated policies. This challenge is compounded by the scarcity of diverse training data (Han et al. 2023) and the dual need for syntactic and semantic fidelity.

To overcome the inherent limitations of the monolithic architecture, we note that the multi-agent system (MAS) (Wang et al. 2024; Hong et al. 2023; Chen et al. 2024), as a computational paradigm where multiple au-

*Corresponding author.

tonomous agents collaborate to solve complex problems, has shown great potential in numerous fields (Hong et al. 2023; Qian et al. 2023; Mandi, Jain, and Song 2024; Zhang et al. 2023; Xu et al. 2023; Park et al. 2023; Boiko, MacKnight, and Gomes 2023). Through task decomposition and collaboration, it can handle complex problems that are intractable for a single agent. However, the idea of applying a MAS to the specific task of automatic ODRL generation has not yet been fully explored. We posit that this architectural paradigm can offer a new perspective for resolving the complexity of "NL-to-ODRL" conversion.

Therefore, this paper designs and implements AgentODRL, an integrated system that synergizes a multi-agent framework with targeted optimization strategies to address these challenges holistically. The system is based on an "Orchestrator-Workers" pattern (Dean and Ghemawat 2008). The core idea is a two-stage process: first, a central Orchestrator Agent analyzes the input's complexity and dispatches the task to specialized Worker Agents—a Rewriter for recursive structures and a Splitter for parallel ones. Second, after the Generator Agent produces an initial ODRL policy, two crucial post-generation strategies are employed to ensure quality: a validator-based iterative correction loop for syntactic accuracy and a LoRA-finetuned reflection mechanism for semantic fidelity.

The main contributions of this paper are as follows:

- **We propose AgentODRL, an integrated framework combining a multi-agent architecture with dedicated syntactic and semantic enhancement strategies.** This design achieves state-of-the-art performance, improving the average grammar and semantic scores across all models by 5.39% and 14.52% respectively when compared to the SOTA strategy (SCR-Enhanced), all while ensuring near-perfect grammatical compliance.
- **We construct and release a new comprehensive benchmark dataset for supporting and evaluating ODRL generation**. We built a dataset of 770 use cases, each with an NL policy for conversion into ODRL. The dataset spans diverse structural complexities—including simple, parallel, and recursive forms—offering a robust foundation for evaluating our system and supporting future research in ODRL generation.

2 Related Work

Automated Generation of ODRL Policies Research on converting NL policies into the ODRL format can be broadly divided into two phases. Early studies primarily relied on ontology extensions to manually or semi-automatically model rules from specific regulations, such as the General Data Protection Regulation (GDPR) or the European Union's Artificial Intelligence Act (EU AI Act) (De Vos et al. 2019; Golpayegani et al. 2024; Esteves, Pandit, and Rodríguez-Doncel 2021). This process requires significant manual intervention and is difficult to apply at scale.

The advent of LLMs introduced a new paradigm, with researchers exploring the use of pre-trained models like BERT for end-to-end knowledge graph construction (Kumar et al. 2020). To enhance generation quality, subsequent studies have introduced methods such as SCR (Mustafa et al. 2025) or employed fine-tuned small models as validators (Han et al. 2023; Sean et al. 2023). These explorations aim to optimize the performance of a single LLM in the end-to-end conversion task.

However, these methods still face core challenges to their practical application. First, high-quality "NL-to-ODRL" parallel corpora are extremely scarce. Related research confirms that data scarcity compels researchers to construct or augment datasets themselves (Han et al. 2023), which fundamentally limits the model's learning capabilities. Second, to circumvent the task's inherent difficulty, existing works generally confine their scope to policy texts with simple logical structures. They fail to systematically address real-world rules containing complex structures, such as parallel or recursive ones, hindering the applicability of their solutions in real-world scenarios.

Multi-Agent Systems for Complex Task Decomposition To overcome the bottlenecks of a single LLM, LLM-based MAS have emerged as a new research paradigm. This paradigm solves problems intractable for a single model by decomposing complex tasks and enabling multiple agents, each playing a specific role, to collaborate and simulate collective intelligence (Wang et al. 2024; Hong et al. 2023; Chen et al. 2024). Currently, this "divide-and-conquer" strategy has demonstrated great potential in diverse domains, including automated software development (Hong et al. 2023; Qian et al. 2023), multi-robot systems (Mandi, Jain, and Song 2024; Zhang et al. 2023), scene simulation (Xu et al. 2023; Park et al. 2023), and scientific research (Hong et al. 2023; Boiko, MacKnight, and Gomes 2023).

However, the application of this paradigm in domains requiring high precision and strict logic, such as data rights management (Calvaresi, Schumacher, and Calbimonte 2020), remains in its nascent stages. The ODRL generation task, in particular, inherently integrates multiple cognitively distinct sub-tasks. This intrinsic complexity poses a fundamental challenge to traditional methods that rely on a single model. In contrast, the multi-agent architecture offers a highly aligned systemic solution through task decomposition and collaboration, revealing the significant potential of using architectural innovation to address the complexity of rule-based generation.

3 Use Case Complexity in Practice

To systematically process NL policies into machine-readable ODRL, we first establish a classification based on their structural complexity. Use cases are categorized into two main types according to their internal structural relationships: **Simple Use Cases** and **Complex Use Cases**. This classification depends on core semantic elements, referred to as "rule information" ($\mathcal{R}_{info}$), which can be formally defined as a tuple:

$$\mathcal{R}_{info} = (P, A, Ac, C_{policy}, C_{context})$$

where P represents the **Party** (Assigner/Assignee); A is the target **Asset**; Ac is the permitted **Action**; C_{policy} is the set of core policy clauses (**Permissions, Prohibitions, and Duties**); and $C_{context}$ is the set of contextual **Constraints**.

3.1 Simple Use Cases

Simple Use Cases are characterized as limited rule information with a monolithic structure. The essence of these use cases is the description of a self-contained policy or a few closely related policies, whose semantics and structure are highly independent without complex dependencies.

> **Example 3.1: Data Access License** *"The DE_Staatstheater_Augsburg, a German cultural organization, manages the dataAPI 'ShowTimesAPI'. This dataAPI holds valuable cultural assets. Policy regulates access to this dataAPI, granting subscribers like 'Regional Newspaper', 'Culture Research Institute', and 'Cultural Platform Bavaria'. Access is restricted to Germany, and usage rights expire on May 10, 2025."*

This example embodies a single, coherent policy. Given their self-contained nature and clear semantics, such simple use cases correspond to a baseline processing path in our proposed workflow. This path directly invokes the **Generator** agent (see Section 4), typically without the need for complex structural preprocessing.

3.2 Complex Use Cases

In contrast to simple cases, the challenge of complex use cases lies not merely in their length but in the intricate structural relationships between their internal policies. We subdivide them into two subtypes: Use Cases with Parallel Structures and Use Cases with Recursive Structures.

Use Cases with Parallel Structures Use Cases with Parallel Structures are characterized by the inclusion of multiple, relatively independent policies within a single use case. Such use cases are commonly found in standards, website terms of service, and comprehensive licensing frameworks.

> **Example 3.2: Tiered Access and Obligations** *"Registered non-commercial drone operators are permitted to download the 'Alpine Geohazard Maps' from SwissTopo for flight planning, although access to the raw data files expires after 72 hours. While using the maps, they have an obligation to credit SwissTopo as the data source in their flight logs and are strictly prohibited from redistributing the original map files to third parties. For commercial companies seeking to integrate the data into proprietary software, access is granted upon payment of a per-square-kilometer fee."*

Example 3.2 clearly contains two parallel policies for different user groups: an "Offer" for non-commercial users and another for commercial companies. To faithfully convert such multifaceted use cases, a critical prerequisite is the logical decomposition of the source text into multiple, independent policy units. In our workflow, this task is undertaken by the **Splitter** agent (see Section 4).

Use Cases with Recursive Structures Use Cases with Recursive Structures are the most complex, featuring explicit cross-clause dependencies where one policy's enforcement is contingent on another. This structure is common in legal acts (e.g., GDPR, CCPA) and formal contracts.

> **Example 3.3: Referential Dependency in Legal Clauses** *(From the Data Security Law of China)*
>
> **Article 48:** *"Whoever, in violation of the provisions of Article 35..., refuses to cooperate..., shall be fined..."*
>
> **Article 35:** *"When public security organs... retrieve data..., relevant organizations... shall cooperate."*

The penalty in Article 48 is contingent on violating Article 35. This dependency, a hallmark of recursive structures, challenges automated conversion. It requires a rewriting stage by our **Rewriter** agent (see Section 4) to resolve the reference, making the policy self-contained for subsequent processing.

4 Methodology

This section details the design and implementation of **AgentODRL**, a MAS for converting NL data permission rules into the ODRL. The conversion is not a single, monolithic task. It inherently involves several distinct cognitive functions, including parsing complex logical structures, segmenting rules based on semantic intent, and generating code under strict syntactic constraints.

To effectively manage this complexity, AgentODRL employs a modular Orchestrator-Workers architecture. This design decouples the overall task, allowing specialized agents to handle specific cognitive functions where they excel. Such an approach enhances the workflow's robustness, interpretability, and maintainability, overcoming the limitations of a single-model approach. The architecture and working principle of this system are illustrated in Figure 1.

4.1 The Orchestrator-Workers Workflow

AgentODRL adopts the "Orchestrator-Workers" pattern, where a central LLM agent—the Orchestrator—acts as a project manager. It analyzes and decomposes tasks, assigning them to specialized "worker" agents.

Orchestrator Agent: Serving as the central cognitive engine of the workflow, this agent receives NL use cases from the user. Its primary function is to rapidly analyze and classify the input based on the complexity criteria defined in the preceding chapter. Based on the classification result, the Orchestrator Agent determines which Worker Agents to invoke and orchestrates an optimal processing path.

Worker Agents: This component comprises a group of specialized agents, each engineered to handle a specific subtask within the conversion process. Based on our analysis of ODRL policy complexity discussed in Section 3, this study implements three core Worker Agents: a **Rewriter Agent** for resolving cross-references in recursive structures, a **Splitter Agent** for decomposing parallel structures, and a **Generator Agent** for converting well-structured rule text into high-quality ODRL policies.

4.2 Worker Agents in Detail

Rewriter Agent The Rewriter Agent addresses the most complex "recursive structure" use cases by performing a "Structure-Preserving Inlining" task. Guided by a structured

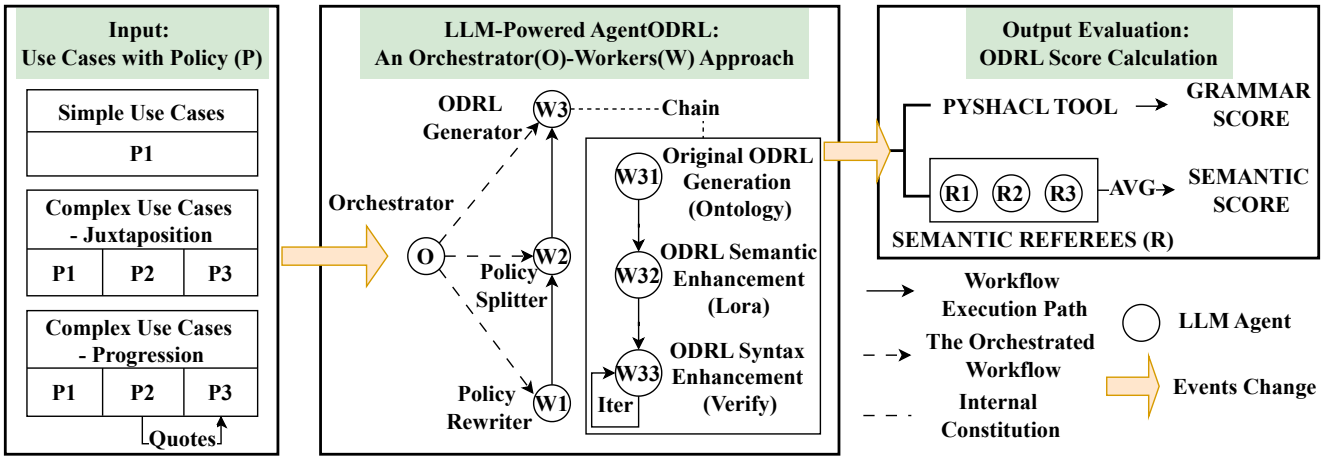

Figure 1: The "Orchestrator-Workers" architecture of AgentODRL. The system takes various policy use cases as input (left panel), processes them through a central workflow orchestrated by an Orchestrator agent that delegates tasks to specialized Worker agents (center panel), and evaluates the final ODRL output using Grammar and Semantic scores (right panel).

prompt, it identifies and resolves both explicit (e.g., references to clause numbers) and implicit (e.g., "Notwithstanding...") cross-references. The agent inlines the content of a referenced clause into the referring clause to eliminate semantic dependency. Crucially, this process preserves the structural separation of the original clauses, thereby simplifying the recursive problem into a set of independent statements prepared for the Splitter Agent.

Splitter Agent The Splitter Agent is designed for "parallel structure" use cases. Its core logic, encoded within a detailed prompt, directs its LLM to segment rules based on fundamental semantic shifts rather than superficial syntax. A new policy unit is initiated only upon a change in: (1) the core asset, (2) the core assigner-assignee relationship, or (3) the fundamental purpose of the policy. Following decomposition, the agent assigns an ODRL type (Agreement, Offer, or Set) via heuristics, ensuring each resulting unit is a self-contained and unambiguous input for the Generator Agent.

Generator Agent The Generator Agent constitutes the core execution unit of the entire workflow. To guarantee the quality of the output, this agent integrates two key optimization strategies.

LoRA-Enhanced Strategy for Semantic Fidelity. To ensure the generated ODRL accurately reflects the intent of the original rule, we propose an innovative, reflection-based semantic optimization process. The core concept is a complementary "generator-validator" model, which pairs a large generative model with a small, specialized "expert validation model" fine-tuned with domain knowledge. We employ Low-Rank Adaptation (LoRA) to fine-tune a lightweight LLM (`Qwen3-4B-Instruct(BF16)`) into a high-precision, low-cost expert agent for semantic extraction. This expert agent extracts key semantic elements (e.g., parties, assets, actions) into a structured "semantic checkpoint" list. The main Generator Agent must then validate its ODRL output against this list, ensuring every semantic point is accurately encoded.

In particular, the LoRA fine-tuning process utilized parameters of r=`16` and $alpha$=`32` on a dataset comprising 2,380 synthetic samples, the construction of which is consistent with the methodology used for creating use cases as detailed in the Section 5. Training was conducted on a single NVIDIA 4090 GPU for 3 epochs. The final model achieved a validation loss ($eval\_loss$=`0.0668`) that was significantly lower than the training loss ($train\_loss$=`0.129`), indicating effective generalization without overfitting and thus ensuring its reliability.

Validator-Based Strategy for Syntactic Correctness. To mitigate potential syntax errors when LLMs directly generate ODRL, we introduce a validator-based "generate-validate-correct" iterative loop. Initially, the Generator Agent produces an ODRL policy. This policy is then submitted to a built-in validator, which is implemented using the PYSHACL library. This validator performs rigorous Shapes Constraint Language (SHACL) (Knublauch and Kontokostas 2017) validation against the ODRL information model and vocabulary. If validation fails, a detailed error report is fed back to the Generator Agent's LLM, prompting it to reflect and revise its output. This loop persists until the policy successfully passes validation or a preset maximum number of attempts is reached, thereby enhancing the syntactic correctness of the final output.

4.3 Workflow Paths

For different use cases, AgentODRL's adaptive workflow dynamically composes agents:

- **Simple Path:** Input → Orchestrator → Generator → Output
- **Parallel Path:** Input → Orchestrator → Splitter → [Units] → Generator → [ODRLs] → Output
- **Recursive Path:** Input → Orchestrator → Rewriter → Splitter → [Units] → Generator → [ODRLs] → Output

5 Dataset Construction

To the best of our knowledge, there exist no dataset for ODRL generation, mainly due to both the complexity of data right policy and ODRL. Given the lack of a suitable benchmark, we also constructed a new dataset tailored for data space scenarios to evaluate ODRL generation capabilities. The process involved two stages: Seed Case Formulation and LLM-based Data Augmentation.

5.1 Stage 1: Seed Case Formulation

We initially formulated 70 seed use cases, categorized by logical structure: simple (40), parallel (20), and sequential-dependency (10). The sources for these cases were diverse, including academic literature (Mustafa et al. 2025), drafts of industry standards[1], and standard license agreements like Creative Commons 4.0. For the structurally complex sequential-dependency cases, we manually curated robust logical templates from legal texts such as the GDPR[2] and the California Consumer Privacy Act (CCPA)[3].

5.2 Stage 2: LLM-based Data Augmentation

We then employed Gemini 2.5 Pro to augment the 70 seed cases, guided by two strict principles: **Core Logic Preservation** and **Contextual Element Transformation**. The former maintained the fundamental policy structure (rule type, action-target binding, constraint categories), while the latter diversified contextual elements (participants, data assets, use cases, etc.) to simulate real-world business variety.

Following this strategy and a subsequent comprehensive manual review, we constructed a high-quality dataset of 770 test cases, comprising **440 simple cases**, **220 parallel-relationship cases**, and **110 sequential-dependency cases**. This dataset establishes a robust foundation for evaluating our model and facilitating future research.

6 Experiments

This section presents two experiments conducted on our constructed dataset. The results demonstrate that our proposed ODRL generation strategy is state-of-the-art on both evaluation metrics, and that the modules of AgentODRL are highly efficient and effective in handling rule conversion.

6.1 Evaluation Metrics and Calculation Methods

To comprehensively and quantitatively evaluate the performance of the proposed AgentODRL system and its various strategies on the NL-to-ODRL conversion task, we designed two core evaluation metrics: the **Grammar Score** and the **Semantic Score**.

[1]China Electronics Standardization Institute (CESI), *General Requirements for Trusted Data Space Operation Rules*.

[2]Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 (*GDPR*). https://eur-lex.europa.eu/eli/reg/2016/679/oj

[3]California Consumer Privacy Act (CCPA) of 2018, Cal. Civ. Code §§ 1798.100 et seq. https://leginfo.legislature.ca.gov/faces/billTextClient.xhtml?bill_id=201720180AB375

Grammar Score The Grammar Score measures whether a generated ODRL policy strictly adheres to the specifications of the W3C ODRL Information Model and Vocabulary. Syntactic correctness is a fundamental prerequisite for a policy's validity. To calculate this score, we employ an automated workflow based on SHACL. The process begins by establishing a comprehensive set of SHACL constraint rules based on the official ODRL standard. Subsequently, each generated ODRL policy is validated against this predefined ruleset to count the number of constraint violations (N_{errors}). The final Grammar Score is computed using the following formula:

$$\text{Score}_{\text{grammar}}(\%) = \left(1 - \frac{N_{\text{errors}}}{N_{\text{constraints}}}\right) \times 100\%$$

where $N_{\text{constraints}}$ is the total number of constraints in the SHACL ruleset, and N_{errors} is the total number of errors reported by the validator.

Semantic Score The Semantic Score quantifies how faithfully a generated ODRL policy reflects the complete intent of the original rule. The assessment of this metric hinges on establishing a "semantic checkpoint list", which comprises all indivisible, independent semantic points from the original rule.

The metric is assessed through an automated workflow based on an LLM Jury. This workflow first involves a dedicated "Identifier" LLM to extract a ground-truth "semantic checkpoint list" from the original NL use case. Subsequently, to ensure objectivity, a "Jury" composed of two independent LLMs ($K = 2$) meticulously compares the generated ODRL policy against the checkpoint list. Each juror assesses how accurately each semantic unit is reflected and assigns a score (S_{ij}). The final Semantic Score is calculated as the average of scores given by all jurors across all semantic units, according to the formula:

$$\text{Score}_{\text{semantic}}(\%) = \frac{1}{N_{\text{units}}} \sum_{i=1}^{N_{\text{units}}} \left(\frac{1}{K} \sum_{j=1}^{K} S_{ij}\right) \times 100\%$$

where N_{units} is the total number of semantic units, K is the number of jurors, and S_{ij} is the score given by the j-th juror for the i-th semantic unit.

6.2 Experiment 1: Evaluating ODRL Generation Strategies

To systematically evaluate the effectiveness of our proposed strategies, we consider the following two baselines:

- **Ontology-Guided Strategy (OGS)**: This approach utilizes the ODRL v2.2 ontology (Iannella et al. 2018) to guide LLMs in generating the ODRL policy.
- **SCR-Enhanced** (Mustafa et al. 2025): Building on OGS, this approach uses predefined rules, sourced from the ODRL W3C recommendation and ontology relationships, to fix errors in the LLM's output and ensure it better aligns with the official specification.

Furthermore, we evaluate our framework progressively by considering two strategies based on whether they incorporate our proposed semantic and syntactic enhancements:

Model	Use Cases	Grammar Score				Semantic Score				Reflections(Avg)
		OGS	SCR-Enhanced	Semantic-Enhanced	AOFP	OGS	SCR-Enhanced	Semantic-Enhanced	AOFP	(For AOFP)
GPT-4.1	ALL Use Cases	82.07	93.08	*93.47*	**99.89**	89.59	92.00	*97.78*	**97.93**	1.29
	Simple Use Cases	81.52	92.70	*93.59*	**99.94**	94.49	95.99	**98.78**	*98.64*	1.22
	Complex - Parallel	81.13	92.23	*92.52*	**99.75**	86.47	90.53	*96.47*	**97.27**	1.62
	Complex - Recursive	86.18	*96.32*	94.91	**99.86**	76.18	*78.97*	**96.40**	**96.40**	0.89
GPT-4.1-mini	ALL Use Cases	82.41	94.83	*95.44*	**99.36**	86.53	88.93	*96.25*	**96.43**	2.12
	Simple Use Cases	83.65	96.14	*96.99*	**99.55**	89.79	91.86	*97.50*	**97.80**	1.44
	Complex - Parallel	79.73	91.85	*92.65*	**98.98**	87.68	90.79	*94.74*	**94.97**	3.49
	Complex - Recursive	82.81	*95.94*	94.81	**99.36**	71.20	73.47	**94.26**	*93.85*	2.08
GPT-4.1-nano	ALL Use Cases	79.77	*88.40*	87.49	**92.01**	50.51	56.23	*65.67*	**72.35**	6.64
	Simple Use Cases	80.02	*89.32*	87.80	**92.13**	55.17	61.03	*70.38*	**76.50**	6.63
	Complex - Parallel	79.18	*87.25*	86.55	**92.26**	49.00	55.34	*61.00*	**69.44**	6.29
	Complex - Recursive	79.98	*89.02*	88.10	**91.07**	34.87	40.40	*56.20*	**61.53**	7.32

Table 1: Complete results for Experiment 1, evaluating ODRL generation strategies across different GPT models and use case complexities. Our AgentODRL's Full Pipeline consistently outperforms both the Ontology-Guide Strategy and SCR-Enhanced methods across all models and complexity levels, with particularly strong gains in semantic accuracy for complex cases and near-perfect grammar scores. For each use case, the best-performing grammar and semantic scores are shown in bold, while the second-best scores are presented in *italic and underlined* format.

- **Semantic-Enhanced Strategy**: This strategy integrates our LoRA-finetuned model to enhance semantic fidelity.
- **AgentODRL's Full Pipeline (AOFP)**: This represents our full, integrated system. It combines the Semantic-Enhanced Strategy with a validator-based strategy for syntactic correctness.

All tests were conducted on the GPT-4.1 series models (GPT-4.1, GPT-4.1-mini, and GPT-4.1-nano). Based on preliminary experiments, the maximum number of reflections was set to 8, a configuration applied to all experiments in this paper. To ensure the robustness of the results and mitigate the effects of randomness, each experimental configuration was run three times, with the final results averaged.

As reported in Table 1, **our AOFP significantly outperforms the existing SCR-Enhanced strategy across all tested dimensions**. Its superiority is manifested in two key areas. First, our strategies significantly enhance both grammar and semantic fidelity. On the "ALL Use Cases" setting, our AOFP framework improves the average grammar score by 5.39% and the average semantic score by a notable 14.52% across the three models when compared to the SCR-Enhanced strategy. The peak improvement is even more pronounced, with the grammar score increasing by up to 7.32% (for GPT-4.1) and the semantic score by up to 28.67% (for GPT-4.1-nano). Second, the syntax strategy effectively ensures ODRL compliance. The "generate-validate-reflect" closed-loop correction mechanism ensures the final output strictly adheres to W3C specifications. Within the AOFP, the grammar scores for all models rose to near-perfect levels (mostly exceeding 99), effectively rectifying syntactic hallucinations from the LLM.

Additionally, **our proposed strategies exhibit a generalizable enhancement effect across models of varying capabilities**. While a model's foundational capabilities determine the performance ceiling (GPT-4.1 $>$ GPT-4.1-mini $>$ GPT-4.1-nano), our strategies notably raise the performance floor of smaller models. For example, when using the Baseline strategy, GPT-4.1-nano scored a mere 34.87 in semantics on recursive structure use cases. However, our AOFP elevates this score to 61.53, achieve a 76.46% improvement.

Moreover, **our strategy exhibits remarkable stability against increasing use case complexity**. As complexity escalates from "Simple" to "Recursive" cases, the performance of the OGS degrades sharply, whereas our AOFP maintains a high degree of stability (e.g., the semantic score for GPT-4.1 only slightly decreased from 98.64 to 96.4). This ability to maintain high performance is also reflected in the strategy's adaptive correction cost. The "Average number of reflections" metric reveals the iterative effort expended by the syntax strategy to achieve compliance. The data show that weaker models and more complex use cases require more reflections (e.g., GPT-4.1-nano averaged 7.32 reflections for recursive cases). This capacity to sustain high performance under pressure while adaptively adjusting its process ensures the strategy's reliability in handling the diverse and complex rules found in real-world scenarios.

6.3 Experiment 2: Orchestrator-Workers Workflow Performance Evaluation

To comprehensively evaluate the overall efficacy of our proposed adaptive Orchestrator-Workers workflow, we designed Experiment 2. This experiment aims to validate its core value in a two-tiered process: first, by forcing all use cases through different fixed workflow paths, we establish the performance "theoretical ceiling" for each complexity level as a benchmark. Second, we test whether the fully automated Orchestrator-Workers workflow can, without human intervention, approach this theoretical ceiling through dynamic analysis and intelligent routing.

To more clearly observe the performance gains, we se-

Use Cases	Workflows	Grammar Score	Semantic Score	Reflections(Avg)	Tokens
Simple Use Cases	ODRL Generator	92.13	76.50	6.63	-
	Splitter → Generator	**94.03**	**85.18**	**5.54**	-
	Rewriter → Splitter → Generator	93.56	84.50	5.54	-
	Orchestrator - Workers	93.45	82.12	5.82	-
Complex - Parallel Structures	ODRL Generator	92.26	69.44	6.29	-
	Splitter → Generator	**93.77**	**84.88**	**6.13**	-
	Rewriter → Splitter → Generator	92.66	83.30	6.44	-
	Orchestrator - Workers	91.11	82.47	6.34	-
Complex - Recursive Structures	ODRL Generator	91.07	61.53	7.32	-
	Splitter → Generator	91.62	77.70	6.61	-
	Rewriter → Splitter → Generator	**93.27**	**82.00**	**6.03**	-
	Orchestrator - Workers	89.45	80.97	6.33	-
ALL Use Cases	ODRL Generator	92.01	72.35	6.64	33856708
	Splitter → Generator	93.62	84.02	5.86	47917323
	Rewriter → Splitter → Generator	93.27	88.07	6.09	49451650
	Orchestrator - Workers	92.56	80.22	6.04	46209570

Table 2: Performance comparison of different workflow paths on the GPT-4.1-nano model. The table details results across specific use case categories and provides an overall summary for all use cases, including token consumption. The results demonstrate the necessity of aligning workflow paths with use case complexity. The best-performing path for each use case category is highlighted in bold.

lected the GPT-4.1-nano model, which exhibited greater potential for improvement, as the execution agent for the Generator module. Concurrently, to maximize the analytical capabilities of the upstream modules, the Orchestrator, Splitter, and Rewriter agents all utilized the more powerful GPT-4.1 model. To ensure the stability and reproducibility of the results, each experimental configuration was run three times, with the final results averaged.

Establishing the Performance Benchmark: The Role of Specialized Agents The data in Table 2 clearly reveals that the optimal processing path is dictated by the task's structural complexity, highlighting the distinct roles of our specialized worker agents.

First, **the Splitter agent, designed for decomposing parallel policy structures, demonstrates universal benefits**. Across all use case categories, the workflow path that incorporates the Splitter before the Generator yields significant performance gains over the "ODRL Generator". For instance, in "Complex - Parallel Structures", the Semantic Score dramatically increases from 69.44 to 84.88.

Second, **the Rewriter agent proves its value in more complex scenarios**. While the full workflow path, which includes the Rewriter, Splitter, and Generator in sequence, enhances performance across the board, its indispensable role is most evident in recursive use cases. Here, it achieves the highest possible Semantic Score of 82.00, effectively resolving the cross-clause dependencies that the baseline model struggles with. These findings validate a core principle: **performance optimization stems from the precise alignment of the processing path with task complexity**.

Orchestrator's Intelligent Decision-Making: Balancing Performance and Efficiency The established benchmarks provide a clear frame of reference for evaluating the Orchestrator's value in automating workflow selection. As shown in Table 2, the Orchestrator-Workers workflow achieves an average Semantic Score of 80.22 across all use cases. This performance is not only significantly better than the ODRL Generator's score of 72.35 but also **demonstrates the effectiveness of the multi-agent collaborative architecture**.

While the Orchestrator's average score is slightly below the theoretical ceilings set by manually-selected, fixed workflows (e.g., 84.02 for the path using the Splitter, and 88.07 for the path using the Rewriter), its primary advantage lies in its efficiency. The Orchestrator intelligently routes tasks, thereby optimizing computational resource usage. The data on token consumption confirms this: the Orchestrator-Workers workflow required 46.2M tokens, which is more economical than forcing all use cases through either the fixed path utilizing the Splitter (47.9M tokens) or the full path initiated by the Rewriter (49.5M tokens). **This demonstrates the Orchestrator's success in automating the "optimal choice" process—dynamically balancing near-optimal performance with significantly lower computational cost, making it a practical and potent solution for real-world applications**.

7 Conclusion

This paper introduced AgentODRL, a novel multi-agent system for converting complex NL policies into the machine-readable ODRL format. Experiments on our newly constructed 770-case benchmark demonstrate that: 1) our integrated generation strategies achieve state-of-the-art grammatical and semantic scores, and 2) the "Orchestrator-Workers" architecture is highly effective at automating the selection of optimal processing paths. These findings establish AgentODRL as a potent solution for the challenges of real-world data policy automation.

Acknowledgments

The work was supported by the National Natural Science Foundation of China (62441230, 62172425), and the Fundamental Research Funds for the Central Universities and the Research Funds of Renmin University of China (22XNKJ04).

References

Akaichi, I.; Slabbinck, W.; Rojas, J. A.; Van Gheluwe, C.; Bozzi, G.; Colpaert, P.; Verborgh, R.; and Kirrane, S. 2024. Interoperable and continuous usage control enforcement in dataspaces. In *The Second International Workshop on Semantics in Dataspaces, co-located with the Extended Semantic Web Conference*.

Boiko, D. A.; MacKnight, R.; and Gomes, G. 2023. Emergent autonomous scientific research capabilities of large language models. *arXiv preprint arXiv:2304.05332*.

Calvaresi, D.; Schumacher, M.; and Calbimonte, J.-P. 2020. Personal data privacy semantics in multi-agent systems interactions. In *International Conference on Practical Applications of Agents and Multi-Agent Systems*, 55–67. Springer.

Chen, S.; Liu, Y.; Han, W.; Zhang, W.; and Liu, T. 2024. A survey on llm-based multi-agent system: Recent advances and new frontiers in application. *arXiv preprint arXiv:2412.17481*.

Dam, T.; Krimbacher, A.; and Neumaier, S. 2023. Policy patterns for usage control in data spaces. *arXiv preprint arXiv:2309.11289*.

De Vos, M.; Kirrane, S.; Padget, J.; and Satoh, K. 2019. ODRL policy modelling and compliance checking. In *International Joint Conference on Rules and Reasoning*, 36–51. Springer.

Dean, J.; and Ghemawat, S. 2008. MapReduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1): 107–113.

Esteves, B.; Pandit, H. J.; and Rodríguez-Doncel, V. 2021. ODRL profile for expressing consent through granular access control policies in solid. In *2021 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, 298–306. IEEE.

Farrell, E.; Minghini, M.; Kotsev, A.; Soler Garrido, J.; Tapsall, B.; Micheli, M.; Posada Sanchez, M.; Signorelli, S.; Tartaro, A.; Bernal Cereceda, J.; et al. 2023. European data spaces-scientific insights into data sharing and utilisation at scale. Technical report, Joint Research Centre.

Golpayegani, D.; Esteves, B.; Pandit, H. J.; and Lewis, D. 2024. AIUP: an ODRL profile for expressing AI use policies to support the EU AI act. In *Joint Proceedings of Posters, Demos, Workshops, and Tutorials of the 20th International Conference on Semantic Systems co-located with 20th International Conference on Semantic Systems (SEMANTiCS 2024)*.

Han, J.; Collier, N.; Buntine, W.; and Shareghi, E. 2023. Pive: Prompting with iterative verification improving graph-based generative capability of llms. *arXiv preprint arXiv:2305.12392*.

Hong, S.; Zhuge, M.; Chen, J.; Zheng, X.; Cheng, Y.; Wang, J.; Zhang, C.; Wang, Z.; Yau, S. K. S.; Lin, Z.; et al. 2023. MetaGPT: Meta programming for a multi-agent collaborative framework. In *The Twelfth International Conference on Learning Representations*.

Iannella, R.; Steidl, M.; Myles, S.; and Rodríguez-Doncel, V. 2018. ODRL Version 2.2 Ontology. W3C Document.

Kellogg, G.; Champin, P.-A.; and Longley, D. 2019. *Json-ld 1.1–a json-based serialization for linked data*. Ph.D. thesis, W3C.

Knublauch, H.; and Kontokostas, D. 2017. Shapes Constraint Language (SHACL). Recommendation REC-shacl-20170720, W3C.

Kotis, K. I.; Vouros, G. A.; and Spiliotopoulos, D. 2020. Ontology engineering methodologies for the evolution of living and reused ontologies: status, trends, findings and recommendations. *The Knowledge Engineering Review*, 35: e4.

Kumar, A.; Pandey, A.; Gadia, R.; and Mishra, M. 2020. Building knowledge graph using pre-trained language model for learning entity-aware relationships. In *2020 IEEE international conference on computing, power and communication technologies (GUCON)*, 310–315. IEEE.

Mandi, Z.; Jain, S.; and Song, S. 2024. Roco: Dialectic multi-robot collaboration with large language models. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 286–299. IEEE.

Meckler, S.; Dorsch, R.; Henselmann, D.; and Harth, A. 2023. The web and linked data as a solid foundation for dataspaces. In *Companion Proceedings of the ACM Web Conference 2023*, 1440–1446.

Mustafa, D. M.; Nadgeri, A.; Collarana, D.; Arnold, B. T.; Quix, C.; Lange, C.; and Decker, S. 2025. From instructions to ODRL usage policies: An ontology guided approach. *arXiv preprint arXiv:2506.03301*.

Nagel, L.; Steinbuß, S.; Otto, B.; and Møller, C. 2023. *IDS Reference Architecture Model 4.0*. Dortmund, Germany: International Data Spaces Association e.V.

Otto, B.; and Jarke, M. 2019. Designing a multi-sided data platform: findings from the international data spaces case. *Electronic markets*, 29(4): 561–580.

Park, J. S.; O'Brien, J.; Cai, C. J.; Morris, M. R.; Liang, P.; and Bernstein, M. S. 2023. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th annual acm symposium on user interface software and technology*, 1–22.

Qian, C.; Cong, X.; Yang, C.; Chen, W.; Su, Y.; Xu, J.; Liu, Z.; and Sun, M. 2023. Communicative agents for software development. *arXiv preprint arXiv:2307.07924*, 6(3): 1.

Sean, W.; Lu, X.; Peter, W.; Faeze, B.; Tianxiao, S.; Daniel, K.; and Yejin, C. 2023. Generating sequences by learning to self-correct. In *Proceedings of The 11th International Conference on Learning Representations (ICLR)*.

Villata, S.; and Iannella, R. 2018. ODRL Information Model 2.2. W3C Recommendation.

Wang, L.; Ma, C.; Feng, X.; Zhang, Z.; Yang, H.; Zhang, J.; Chen, Z.; Tang, J.; Chen, X.; Lin, Y.; et al. 2024. A survey on

large language model based autonomous agents. *Frontiers of Computer Science*, 18(6): 186345.

Xu, Z.; Yu, C.; Fang, F.; Wang, Y.; and Wu, Y. 2023. Language agents with reinforcement learning for strategic play in the werewolf game. *arXiv preprint arXiv:2310.18940*.

Zhang, H.; Du, W.; Shan, J.; Zhou, Q.; Du, Y.; Tenenbaum, J. B.; Shu, T.; and Gan, C. 2023. Building cooperative embodied agents modularly with large language models. *arXiv preprint arXiv:2307.02485*.