Gypscie-KG: Building a Logic-Based Approach for Knowledge Graph Data Integration View in ML Systems

Gabriela Moraes[1], Fabio Porto[1], Federico Ulliana[2], Jean-François Baget[2], Michel Leclère[2], Pierre Bisquert[2], Bernardo Gonçalves[1], Patrick Valduriez[1,2]

[1]National Laboratory for Scientific Computing (LNCC) – Petrópolis – RJ – Brazil

[2]LIRMM, Inria, Univ Montpellier, CNRS, INRAE – Montpellier – France

gmoraes@posgrad.lncc.br

Abstract. *This paper presents Gypscie-KG, a system that integrates heterogeneous machine learning (ML) data into a knowledge graph (KG) using logic-based rules to enable semantic queries and reasoning. It also explores the use of ChatGPT to support relational-to-graph schema mapping, demonstrating AI's potential in declarative data integration for complex ML systems.*

1. Introduction

The sheer volume of available data and its continuous generation across scientific, governmental, and industrial domains, combined with increasing computational power, has enabled the development of machine learning (ML) applications that support decision-making in areas such as medical diagnosis, weather forecasting, and fault detection [Toma and Bezemer 2024]. These applications rely on heterogeneous, often streaming, data sources that require pre-processing. Managing the ML lifecycle has led to the emergence of ML systems designed to abstract this complexity [Schlegel and Sattler 2023, da Silva et al. 2019, Boehm et al. 2016]. While existing platforms such as MLFlow [Chen et al. 2020] and AWS SageMaker [Nigenda et al. 2022] provide valuable training and operational support, they focus primarily on artifact management and lack an integrated view that unifies historical, domain-specific, predictive data and ML metadata. This limitation impairs reasoning and knowledge extraction in complex domains. To address this, a declarative approach that integrates relevant data into a knowledge graph (KG) was proposed in [Moraes 2024], making implicit relationships explicit and enabling declarative queries [Ulliana 2021]. This semantic layer is represented through rule-based formalisms such as First-Order Logic (FOL). In this context, we present Gypscie-KG, an ML system that combines data integration, rule-based reasoning, and prediction services to provide semantic access to domain knowledge. The system is proposed not only to integrate heterogeneous ML data into a KG but also to explore the use of logic-based declarative techniques to enable reasoning and semantic querying. In addition to describing the architecture and services offered, this work focuses on evaluating the mapping process between relational schemas and KGs, one of the main challenges of semantic integration. The main contribution consists of investigating the use of ChatGPT as an auxiliary tool in this process, evaluating its ability to automatically generate mappings from relational schemas. The results generated by the model are compared to a reference manually defined by the authors themselves, allowing us to analyze the feasibility of using large-scale language models (LLMs) to support declarative data integration in complex ML systems.

2. Use Case

Gypscie is an ML system designed to support the full ML lifecycle, encompassing data ingestion, curation, model training, and deployment of predictive services [Porto and Valduriez 2022]. Gypscie-KG, on the other hand, refers specifically to the KG component of Gypscie, which provides a semantic, integrated view of all data and metadata managed by the system. Throughout this paper, we use *Gypscie* to refer to the full system, and *Gypscie-KG* to denote the KG that enables integrated querying, traceability, and analysis of the data and ML artifacts involved. Gypscie-KG was developed to support activities such as those of Sue, a meteorologist at the Rio de Janeiro Control and Monitoring Center (COR), who monitors rainfall events and uses a web interface connected to the system to access forecasts and inform decision-making to mitigate the effects of extreme weather. The system integrates real-time data from various sensors, which are preprocessed and stored in a data lake, with all steps and metadata registered in the Gypscie catalog. These curated data serve as input to ML models scheduled to run according to frequencies defined by Sue. Bob, an ML engineer, uses Gypscie to train models with historical data, assess data quality, and analyze training performance and inference metrics. The key feature of the system, Gypscie-KG, provides an integrated view of all data supporting the rainfall forecasting service, enabling declarative queries for both Sue and Bob, and facilitating new functionalities and exploratory analyses.

3. Preliminaries

We introduce the concept of KG as a data model that captures entities and relationships, followed by a rule-based framework for data integration and expression of domain rules.

3.1. Knowledge Graph

We adopt the property graph model for KGs, defined as $G = (V, E, L, P, U)$, where V are nodes (entities), E edges (relationships), L labels (concept identifiers), P properties (key-value pairs), and U literal values [Hogan et al. 2021]. Both nodes and edges can have labels and properties. Figure 1 shows a weather forecast KG snippet used in the COR application.

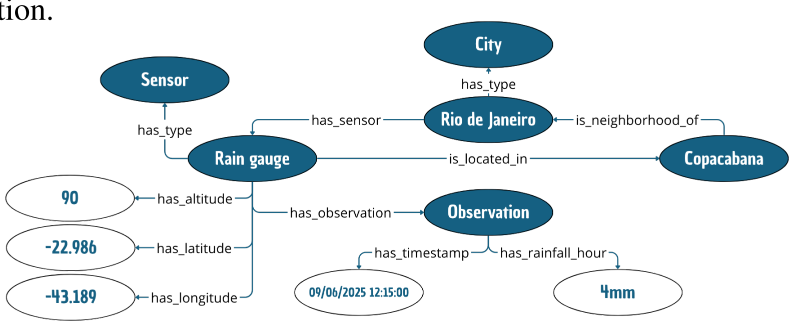

Figure 1. Weather forecast property graph snippet.

3.2. Logic-Based Framework for Data Integration

We adopt the InteGraal framework [Baget et al. 2023], developed within the BOREAL project, to enable semantic integration and reasoning over heterogeneous data sources. InteGraal unifies data from relational databases, SPARQL endpoints, RDF datasets, and in-memory structures into a single knowledge base through rule-based mappings that align source schemas with an ontological vocabulary. Ontologies, in this context, are formal structures that define a set of concepts and relationships in a specific domain, enabling the sharing and reuse of knowledge in a standardized and semantic manner. By aligning data with an ontological vocabulary, it becomes possible to integrate diverse sources and reason about them in a unified manner. This approach allows high-level queries over the KG to be translated into source-specific queries, abstracting data heterogeneity. Reasoning is performed using Datalog-like rules, supporting inference and query answering over federated data sources as if they were a unified dataset.

To illustrate, consider a knowledge base integrating observational weather data. The ontology includes the class `WeatherObservation` and properties such as `hasTemperature` and `observedAt`. The source is a relational table `observations(Sid, Time, Temp)`. The following mapping rules define the integration:

```
WeatherObservation(Obs) :- observations(Sid, Time, Temp),
IRI("http://example.org/obs/", Sid, "_", Time, Obs).

hasTemperature(Obs, Temp) :- observations(Sid, Time, Temp),
IRI("http://example.org/obs/", Sid, "_", Time, Obs).

observedAt(Obs, Time) :- observations(Sid, Time, Temp),
IRI("http://example.org/obs/", Sid, "_", Time, Obs).
```

These rules map each observation record to an RDF individual, associating it with temperature and timestamp values. This enables ontology-driven queries while abstracting the original database schema.

4. The Gypscie System

As mentioned in Section 2, Gypscie is an ML system that supports the full ML lifecycle [Porto and Valduriez 2022]. Figure 2 depicts the complete proposal of the system, including its main components and data flows. As illustrated, Gypscie provides a web interface for ML engineers, model developers, and data engineers to interact during data preparation and model development stages. External systems can register datasets through the Gypscie API, including sensor data (e.g., rainfall, wind) and domain data (e.g., topography). Engineers use Gypscie to build dataflows that implement ML services, which are scheduled to run on registered computational environments, such as a supercomputer or local cluster. The final ML applications deliver information to end users — in our case, weather forecasts consumed by Sue. Access to the data managed by Gypscie is provided through queries to a KG, Gypscie-KG, which offers an integrated view of the metadata and data produced and consumed by system artifacts. This component, represented in Figure 2 as the *Knowledge graph,* is the main focus of this article. The remaining architectural elements have been presented in previous works [Porto and Valduriez 2022] and are included here to provide context for the full system operation.

4.1. Gypscie Data Model

The Gypscie data model [da Silva et al. 2019] focuses on managing metadata for key artifacts such as ML models, datasets, functions, and dataflows, including detailed provenance information for all executions. Importantly, Gypscie does not store the actual application data directly; instead, it orchestrates metadata about heterogeneous data sources, which include domain datasets encompassing sensor observations, model predictions, and historical static data. These data sources follow diverse relational schemas and formats,

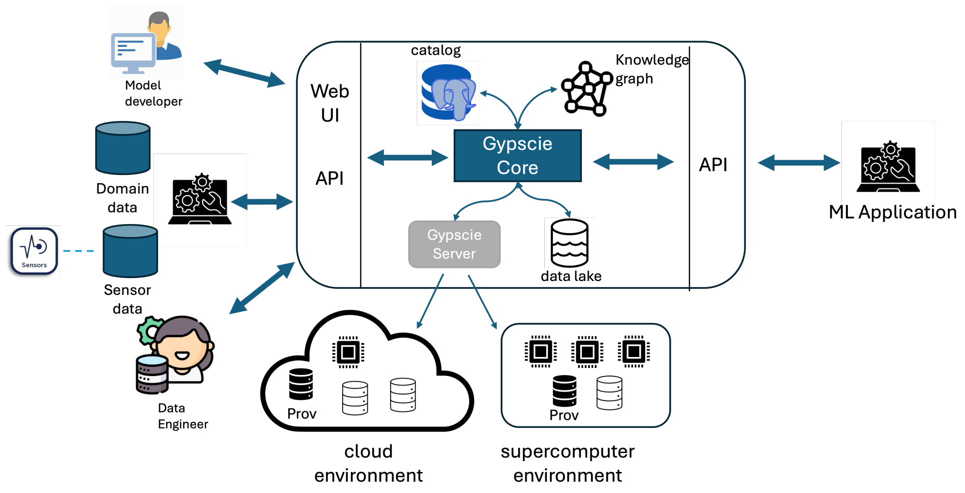

Figure 2. Gypscie Overview.

reflecting their heterogeneity. Gypscie's role is to provide a unified metadata catalog that enables tracking, integration, and management of such diverse artifacts and their associated data without duplicating the original data.

5. Methodology

This section describes the methodology for integrating heterogeneous data and metadata from ML applications using Gypscie-KG, as exemplified in the weather forecast use case.

5.1. Gypscie Knowledge Graph

The Gypscie-KG extends the property graph model to comprehensively represent all artifacts, data, and metadata managed by Gypscie, highlighting the heterogeneity of data sources. For instance, the `Model` nodes correspond to metadata about ML models maintained by the Gypscie catalog, whereas `Time Series` nodes represent actual data stored externally in files, such as rainfall and temperature time series, reflecting different domain-specific types. Additionally, the KG incorporates dataflows and transformed datasets as distinct entities, acknowledging that while `Dataset` nodes capture metadata, the actual data instances linked to these datasets are external and must be integrated from heterogeneous sources. Nodes model domain entities (e.g., weather stations), ML lifecycle processes, provenance, and computational environments. Edges encode relationships such as domain links, properties, types, provenance, and service executions.[1] Figure 3 illustrates how domain entities, artifacts, and services are integrated to support ML applications across heterogeneous data sources.

5.2. Defining Global-as-View Knowledge Graph

The KG adopts a Global-as-View (GaV) approach, where classes and properties are defined as logical views over the Gypscie relational schema. Declarative rules integrate metadata, domain data, and ML artifacts, capturing provenance, data workflows, and spatio-temporal relations. This enables unified, logic-based queries over the entire ML lifecycle. For example, consider the following rule defining a `Model` class as a view over a relational table `models(ID, Name, CreatedAt)`:

[1]For brevity and in adherence to the short paper format, the complete specification of the KG schema and mapping rules is not presented in this paper.

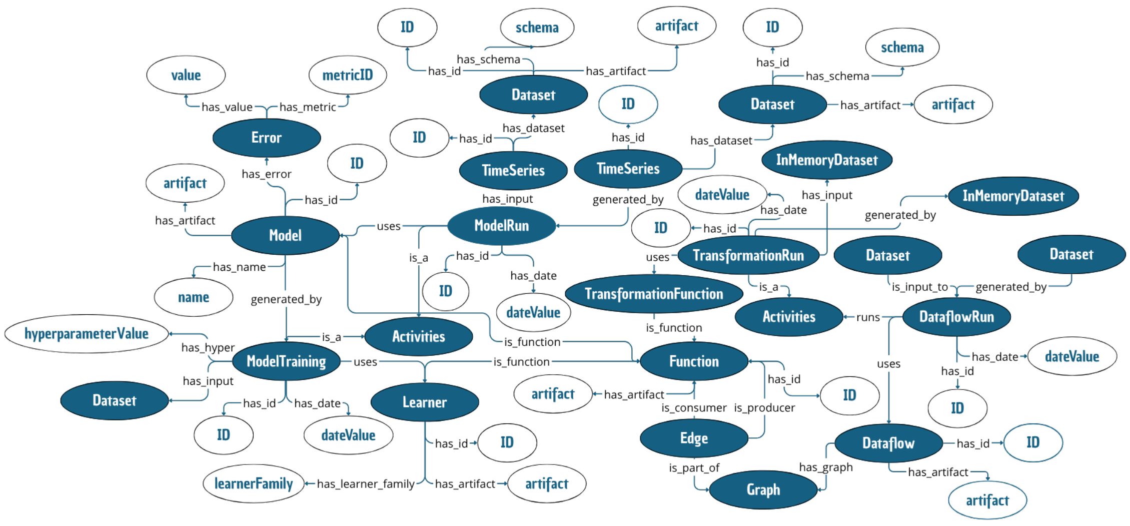

Figure 3. Gypscie Knowledge Graph fragment.

```
Model(M) :- models(ID, Name, CreatedAt),
IRI("http://gypscie.org/model/", ID, M).
```

This rule states that each row in the `models` table corresponds to an instance of the `Model` class in the KG, identified by a unique IRI. Similar rules define properties and relationships, enabling the integration of heterogeneous data sources into a unified semantic layer.

In addition to structural mappings, GaV rules can also define derived concepts based on logical conditions. For instance, the rule below defines alerts for extreme rainfall based on associated time series and intensity thresholds:

```
Alert_extreme_rainfall_KG(Date, Time, Intensity) :-
model_run_KG(URIMR), has_timeseries_KG(URIMR, URITS),
has_rainfall_KG(URITS, Intensity), has_date_KG(URITS, Date),
has_time_KG(URITS, Time), Intensity > 50.
```

Such high-level rules illustrate the expressive power of the GaV approach, enabling the semantic representation of application-level concepts directly within the KG.

5.3. Defining Mappings

Mappings associate nodes and edges of the KG with conjunctive expressions whose terms correspond to relational views over diverse data sources. In Gypscie-KG, these data sources include the system catalog as well as tabular files containing domain data. The mappings are expressed as logical rules with predicates representing the relational views, the actual SQL queries are used in the implementation of these predicates when accessing the underlying data sources.

5.3.1. Using ChatGPT to Generate Mappings

We used *ChatGPT* (GPT-4.5, April 2024) to generate mappings between the data sources' schemas, specifically the Gypscie catalog schema and the external domain data schemas, and the KG. Prompts included syntax constraints (binary predicates, lowercase names) and examples of logical assertions, such as:

```
assert mortal(X) :- human(X).
assert man(socrates).
```

The relational schema was then provided, including tables like `Model`, `Model_run`, and `Dataset`, along with KG elements such as `Model_KG` and `Observation_KG`. ChatGPT generated URI construction rules and logical mappings for nodes and properties, like:

```
assert model_run_kg(URIMR) :-
modelrun(PK_MORU, _, _, _), uri_model_run(PK_MORU, URIMR).

assert model_run_has_date_kg(URIMR, StartDate) :-
modelrun(PK_MORU, StartDate, _, _), uri_model_run(PK_MORU, URIMR).

assert model_run_has_model_kg(URIMR, URIModel) :-
modelrun(PK_MORU, _, _, FK_MODE), uri_model_run(PK_MORU, URIMR),
uri_model(FK_MODE, URIModel).

assert model_run_has_dataset_kg(URIMR, URIDS) :-
employed_dataset(_, FK_DATASET, FK_MODEL_RUN,_),
uri_model_run(FK_MODEL_RUN, URIMR), uri_dataset(FK_DATASET, URIDS).
```

This syntax does not use prefixes or variable markers because it follows the `.dlgp` convention, where predicate names refer to KG concepts or properties by naming convention, and variables are identified by uppercase letters.

5.3.2. Manual Mappings

Manual mappings were created by the authors to link KG elements to both the Gypscie catalog schema and the external dataset schemas. These mappings define nodes such as `Model_KG`, `Dataset_KG`, and `Model_run_KG`, along with properties like `has_id_KG`, `has_artifact_KG`, and relationships including `has_model_KG` and `has_dataset_KG`. Temporary predicates (e.g., `Model_temp_KG`) organize intermediate data for complex mappings. For example, the rule below associates a model execution with its corresponding model in the KG by joining relational data IDs. These manual mappings, crafted by the authors based on their domain knowledge of the Gypscie system, served as a baseline for comparison with the mappings generated by ChatGPT.

```
has_model_KG(URIMR, URIModel) :-
Model_run_temp_KG(URIMR, IDModel), Model_temp_KG(URIModel, IDModel).
```

In addition, mappings were defined for domain data such as rainfall time series, which are associated with spatial regions and temporal intervals, following a specific schema. The example below illustrates how rainfall time series data are linked to KG nodes representing these entities:

```
rainfall_time_series_KG(URITS) :-
Time_series_temp_KG(URITS, IDTS), rainfall_dataset(IDTS).

has_region_KG(URITS, URIR) :-
Time_series_temp_KG(URITS, IDTS), region(IDTS, RegionID),
uri_region(RegionID, URIR).

has_time_interval_KG(URITS, StartTime, EndTime) :-
```

```
Time_series_temp_KG(URITS, IDTS), time_interval(IDTS, StartTime, EndTime).
```

5.4. Querying the Knowledge Graph

This section presents example queries over the KG integrating metadata, provenance, and ML data, generated with ChatGPT.

Q1) Retrieve the input dataset name for model run ID = 10:

```
input_dataset_name_for_model_run_10(Name) :-
employed_dataset(_,PK_DATA,"10","I"), uri_model_run("10", URIMR),
uri_dataset(PK_DATA, URIDS),model_run_has_dataset_kg(URIMR, URIDS),
dataset_has_name_kg(URIDS, Name).
```

Q2) Retrieve observations linked to the input dataset of model run ID = 10:

```
observations_for_model_run_10(URIOBS) :-
uri_model_run("10",URIMR), employed_dataset(_, PK_DATA, "10", "I"),
uri_dataset(PK_DATA, URIDS), model_run_has_dataset_kg(URIMR,URIDS),
time_series(IDTS, PK_DATA, _, _, _, _), uri_time_series(IDTS, URITS),
has_dataset_kg(URITS, URIDS), observation(IDTS, _, _, Time, _),
uri_observation(IDTS, Time, URIOBS), observation_kg(URIOBS).
```

Q3) Identify datasets used in training model "ConvLSTM":

```
datasets_used_in_training_convlstm(DatasetName) :-
model_kg(URIModel), model_has_artifact_kg(URIModel, "CONVLSTM"),
model_training_kg(URIMOTR), has_model_kg(URIMOTR, URIModel),
employed_dataset(_, FK_Data, _, URIMOTR, Role),
(Role = "I"; Role = "O"), uri_dataset(FK_Data, URIDS),
dataset_kg(URIDS), dataset_has_name_kg(URIDS, DatasetName).
```

6. Discussion

We evaluated ChatGPT's capability to generate mappings between the Gypscie-KG and its relational schema, comparing it with manual methods. Initial outputs contained redundant computations due to repeated predicate executions, as shown in Section 5.3.1, leading to inefficiencies with large datasets. To optimize, ChatGPT was guided to introduce intermediate predicates (e.g., `model_training_view`) that aggregate data into reusable logical structures, similar to materialized views, significantly improving performance. These intermediate predicates support multiple KG assertions and reduce query overhead, as exemplified by the rules defining model training, execution environment, and related properties. Below is an example of output generated after optimization:

```
model_training_view(PK_MOTR, URI_MOTR, URI_MODEL, URI_EXEN, ModelName,
ExenName, ExenURI) :-
model_training(PK_MOTR,_,_,FK_MODE,FK_EXEN), model(PK_MODE,ModelName),
FK_MODE = PK_MODE, execution_environment(PK_EXEN, ExenName, ExenURI),
FK_EXEN = PK_EXEN, uri_model_training(PK_MOTR, URI_MOTR),
uri_model(PK_MODE,URI_MODEL),
uri_execution_environment(PK_EXEN,URI_EXEN).

assert model_training_kg(URI_MOTR) :-
model_training_view(_,URI_MOTR,_,_,_,_,_).

assert has_model_kg(URI_MOTR, URI_MODEL) :-
model_training_view(_, URI_MOTR, URI_MODEL, _, _, _, _).
```

```
assert has_execution_environment_kg(URI_MOTR, URI_EXEN) :-
model_training_view(_, URI_MOTR, _, URI_EXEN, _, _, _).
```

Despite requiring expert refinement, ChatGPT effectively approximated the quality of manual mappings, demonstrating its value as a productivity aid in KG construction. In addition to the mappings, we evaluated the quality of the queries generated by ChatGPT, as presented in Subsection 5.4. The queries were syntactically correct and captured the intended semantics, demonstrating the model's ability to follow structural and naming constraints provided in the prompts (e.g., use of binary predicates and consistent variable naming).

7. Related Work

Prior work on automated mapping from relational data to KGs is growing. Zhao et al. [Zhao et al. 2023] proposed *Rel2Graph* for transforming relational schemas into property graphs. Angles et al. [Angles et al. 2020] introduced schema-dependent and schema-independent mappings from RDF to property graphs. Recent studies explore LLMs for graph construction: Kim et al. [Kim et al. 2023] presented *KG-GPT* for reasoning over KGs, and Zhang et al. [Zhang et al. 2025] showed LLMs generating graph schemas. Our approach combines LLM-assisted mapping with manual oversight to enable efficient KG construction in domain-specific contexts.

8. Conclusion

This work demonstrated that ChatGPT can assist in generating logic-based mappings between relational data and a KG in complex domains. With proper guidance, its outputs approached manual mappings. Optimization strategies, such as intermediate predicates, improved query performance. Results indicate that large language models are valuable tools for KG construction when combined with domain expertise.

References

Angles, R., Thakkar, H., and Tomaszuk, D. (2020). Directly mapping rdf databases to property graph databases. *IEEE Access*, 8:90844–90861.

Baget, J.-F., Bisquert, P., Leclère, M., Mugnier, M.-L., Pérution-Kihli, G., Tornil, F., and Ulliana, F. (2023). Integraal: a tool for data-integration and reasoning on heterogeneous and federated sources. In *BDA 2023 - 39e Conférence sur la Gestion de Données – Principes, Technologies et Applications*, Montpellier, France.

Boehm, M., Dusenberry, M. W., Eriksson, D., Evfimievski, A. V., Manshadi, F. M., Pansare, N., Reinwald, B., Reiss, F. R., Sen, P., Surve, A. C., and Tatikonda, S. (2016). Systemml: Declarative machine learning on spark. *Proceedings of the VLDB Endowment*, 9(13):1425–1436. 12 pages.

Chen, A., Chow, A., Davidson, A., DCunha, A., Ghodsi, A., Hong, S. A., Konwinski, A., Mewald, C., Murching, S., Nykodym, T., Ogilvie, P., Parkhe, M., Singh, A., Xie, F., Zaharia, M., Zang, R., Zheng, J., and Zumar, C. (2020). Developments in mlflow: A system to accelerate the machine learning lifecycle. In *Proceedings of the Fourth International Workshop on Data Management for End-to-End Machine Learning*, DEEM '20, New York, NY, USA. Association for Computing Machinery.

da Silva, D. N. R., Simões, A., Cardoso, C., de Oliveira, D. E. M., Rittmeyer, J. N., Wehmuth, K., Lustosa, H., Pereira, R. S., Souto, Y. M., Vignoli, L. E. G., Salles, R., de S. C. Jr, H., Ziviani, A., Ogasawara, E. S., Delicato, F. C., de Figueiredo Pires, P., da C. Pereira Pinto, H. L., Maia, L., and Porto, F. (2019). A conceptual vision toward the management of machine learning models. In Panach, J. I., Guizzardi, R. S. S., and Claro, D. B., editors, *Proceedings of the ER Forum and Poster & Demos Session 2019 on Publishing Papers with CEUR-WS co-located with 38th International Conference on Conceptual Modeling (ER 2019), Salvador, Brazil, November 4, 2019*, volume 2469 of *CEUR Workshop Proceedings*, pages 15–27. CEUR-WS.org.

Hogan, A., Blomqvist, E., Cochez, M., d'Amato, C., de Melo, G., Gutiérrez, C., Kirrane, S., Labra Gayo, J. E., Navigli, R., Neumaier, S., Ngonga Ngomo, A.-C., Polleres, A., Rashid, S. M., Rula, A., Schmelzeisen, L., Sequeda, J. F., Staab, S., and Zimmermann, A. (2021). *Knowledge Graphs*. Number 22 in Synthesis Lectures on Data, Semantics, and Knowledge. Springer.

Kim, J., Kwon, Y., Jo, Y., and Choi, E. (2023). Kg-gpt: A general framework for reasoning on knowledge graphs using large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 9410–9421, Singapore. Association for Computational Linguistics.

Moraes, G. (2024). Integrando observações e predições em grafos de conhecimento ontologicamente fundamentados. Master's thesis, Laboratório Nacional de Computação Científica, Petrópolis-RJ, Brazil.

Nigenda, D., Karnin, Z., Zafar, M. B., Ramesha, R., Tan, A., Donini, M., and Kenthapadi, K. (2022). Amazon sagemaker model monitor: A system for real-time insights into deployed machine learning models. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '22, page 3671–3681, New York, NY, USA. Association for Computing Machinery.

Porto, F. and Valduriez, P. (2022). Data and machine learning model management with gypscie. In *CARLA 2022 - Workshop on HPC and Data Sciences meet Scientific Computing*, pages 1–2, Porto Alegre, Brazil. SCALAC.

Schlegel, M. and Sattler, K.-U. (2023). Management of machine learning lifecycle artifacts: A survey. *SIGMOD Rec.*, 51(4):18–35.

Toma, T. R. and Bezemer, C.-P. (2024). An exploratory study of dataset and model management in open source machine learning applications. In *Proceedings of the ACM Conference*. Association for Computing Machinery.

Ulliana, F. (2021). *Rule-based Languages for Reasoning on Data: Analysis, Design and Applications*. PhD thesis, Université de Montpellier, Montpellier, France.

Zhang, B., He, Y., Pintscher, L., Peñuela, A. M., and Simperl, E. (2025). Schema generation for large knowledge graphs using large language models. *CoRR*, abs/2506.04512.

Zhao, Z., Liu, W., French, T., and Stewart, M. (2023). Rel2graph: Automated mapping from relational databases to a unified property knowledge graph. *CoRR*, abs/2310.01080.