Mining for Meaning: Ontology-Aware Process Mining Methods Through Knowledge Patterns

Riley Moher and Michael Gruninger

Department of Mechanical and Industrial Engineering, University of Toronto, Ontario, Canada M5S 3G8

Abstract. Process mining has emerged as a critical practice for understanding business processes through data-driven analysis. Practitioners necessarily apply diverse process and domain knowledge to guide their analyses. However, these practices are often ad-hoc and informal, failing to formalize the process knowledge being applied. While various formal methods including temporal logic have been applied to process mining, they fail to acknowledge fundamental ontological commitments of processes. Formal ontology for processes, on the other hand, are difficult to integrate into a process mining pipeline, lacking a data-driven grounding. We thus introduce process meaning patterns as a formal declarative framework to capture process knowledge being applied in process mining, based on first order logic ontology patterns. We demonstrate our framework's ability to semi-automatically infer process knowledge motivated by real applications from Volvo IT Belgium. This paper also accompanies a preliminary implementation of the framework using Python and Datalog.

Keywords: process mining · ontology · logic programming · datalog

1 Introduction

Business processes are a nuanced and significant value engine for enterprises. Process mining [1, 2] has enabled a bottom-up and data-driven approach to understanding and analyzing these processes, leveraging event logs to uncover, understand, and optimize complex business processes across diverse domains. Given the accelerating volume of enterprise process data, efficient and precise interpretation of this data has become a critical opportunity. However, project budgets, rushed timelines, and heterogeneity of data introduce significant constraints in practice, leading to a reliance on ad-hoc and informal methodologies [3, 4]. These constraints limit the depth of process analysis and results in critical process knowledge being opaque and unsuitable for re-use or diagnosis.

Formal methods including upper ontologies [5–7] promise a more formal and structured approach to handle these issues, employing formal logic to represent and reason about fundamental process concepts. However, while these ontologies are expressive and well-formalized, they lack grounding in real enterprise data and lack clear applications for process mining pipelines. While applications of formal logic [8–10], knowledge graphs [11, 12], and declarative models [8, 9] have been applied to process mining, they focus on narrow, task and model-specific reasoning and lack comprehensive

verification. While they employ formal methodologies, they fail to acknowledge foundational process distinctions such as events versus states, event subsumption, or point versus interval-based semantics. While these methods provide benefits like structured vocabularies [13, 14] and more meaningful data access [15], they fail to capture fundamental ontological commitments needed for robust and reusable process understanding.

To address these limitations, we introduce a framework for formalizing process knowledge applied throughout the process mining lifecycle to enable transparency and replicability. This is accomplished through what we call "process meaning patterns": first-order logic ontology patterns that correspond to common verification and inference steps in the process mining lifecycle.

The process meaning pattern framework (PMP) demonstrated in this paper is a step towards a more meaningful and practical approach to understanding processes, which can be understood in terms of two goals. Firstly, the goal of ontology-aware process mining: to provide a means to formalize process knowledge applied throughout the process mining lifecycle to enable transparency and replicability. Secondly, the goal of data-aware process ontology: to have process ontologies informed by both the available data and the needs of the domain, striking a balance between accurately reflecting complex realities and providing useful abstractions for specific applications. In this paper, we focus on our contributions towards the former goal.

The contributions of this paper include: (1) a presentation of the process meaning pattern framework as a means to implement ontology-aware process mining. (2) Providing formal specifications of process meaning patterns based on real-world applications and data. (3) Providing a ready-to-use python-datalog implementation of the framework demonstrated on a real-world application from Volvo IT Belgium.

The paper is structured as follows: Section 2 expands on the motivation through real-world process mining challenges, focusing on process data quality and semantic heterogeneity. Section 3 critiques key literature from business process management, process mining, database theory, and formal ontology. Section 4 introduces the process meaning pattern methodology, including first-order logic axiomitizations and demonstrates this methodology using our Python and Datalog-based implementation applied to real enterprise data and use cases from Volvo. Finally, Section 5 situates the paper within process mining and knowledge representation research, including notable future research directions.

2 Process Meaning Challenges

Process mining transforms event logs and enterprise data into actionable process insights. However, it relies on implicit domain assumptions while lacking formal or standardized means of documentation, at the cost of reproducibility and verifiability. These challenges are further amplified by data heterogeneity and the lack of standardized handling of process data quality issues.

Process Data Quality Issues refer to event logs being noisy and incomplete, requiring pre-processing steps including data cleaning, filtering, and transformations. These steps are frequently viewed as secondary to core tasks such as process discovery, which constructs process models from event logs, or conformance checking, which compares

logs' observed behaviour against expected models [1]. Recent deep learning approaches treat noise implicitly [16], while more traditional techniques assume clean data [17], obscuring valuable process knowledge embedded in these pre-processing tasks. Process mining is an iterative process: insights from discovery may prompt adjustments to data handling, and these pre-processing steps should be explicitly modelled rather than handled in an ad-hoc manner. Capturing event relationships, state transitions, and ordering constraints formally would make these steps modular and reusable, enhancing adaptability across mining tasks.

Recent work introduces imperfection patterns as a taxonomy of process mining data quality issues, outlining their impact, detection, and resolution strategies [3]. While potentially useful in practice, these patterns lack formal grounding and overlook the role of process knowledge in resolving data issues. For instance, the unanchored event pattern covers issues where timestamp formats vary (e.g., `MM/DD/YYYY` vs. `DD/MM/YYYY`). This is a syntactic issue simply requiring format standardization. By contrast, the *homonymous label* pattern describes issues where identical event labels refer to distinct behaviours, an issue of semantic heterogeneity. For example, a hospital log with several consecutive `Triage Assessment` events may in-fact represent distinct actions, such as an initial assessment and a follow-up. Resolving this issue requires reconciliation of process constraints and behaviours, not simply renaming labels or changing formats. Without formalizing such process knowledge, ad-hoc fixes obscure important semantics, reducing transparency and reuse. A structured, ontology-driven approach to data quality handling would improve consistency, verifiability, and integration within the process mining pipeline.

Semantic Heterogeneity of Event Logs is a result of the central artifact of process mining, the event log, being commonly recorded in the extensible event stream (XES) format [18]. XES is intentionally flexible, defining events as "an atomic granule of activity that has been observed,". This flexibility introduces semantic heterogeneity when representing diverse process entities such as status values, activities, resources, and occurrences. The Business Process Intelligence Challenges (BPIC) [1], widely used as process mining benchmarks [4, 19], highlight these challenges. For instance, BPIC 2012 poses the challenge to "Identify which decisions have greater influence on the process flow". Addressing this challenge requires understanding complex process constructs including decisions, control flow, and causality — concepts not explicitly defined in event logs. Moreover, event data quality and granularity vary significantly; across ten BPIC datasets, counts of unique activity labels range from 4 to 624 [2].

The XES lifecycle extension classifies events with "lifecycle transitions" such as start, complete, schedule, or abort. However, real-world logs often lack explicit lifecycle tagging, making interpretation unclear. For example, BPIC 2017 explicitly tags events like "Complete Application" with start and complete transitions, while BPIC 2013's activity labels model status values (ex: "Accepted," "Queued"), with lifecycle transitions refining them (ex: "Awaiting Assignment"). Without standardization or on-

[1] `tf-pm.org/competitions-awards/bpi-challenge/`

[2] We provide our own quantitative analysis of these datasets at `github.com/riley-momo/ProcessMeaningPatternsPython/tree/main/notebooks`

tological grounding, event log interpretation remains a key challenge, impacting consistency and integration of knowledge within the process mining pipeline.

3 Related Work

Many recent works have attempted to formalize process mining, including through ontologies. However, they face limitations in reasoning, verification, and scope, often tailoring logic to specific applications while neglecting fundamental ontological commitments. Moreover, existing process ontologies lack a practical grounding and are thus difficult to integrate into process mining pipelines.

Event Knowledge Graphs (EKGs) [20] extend process mining by representing multiple process perspectives based on objects involved in the processes like purchase orders or shipments. This is accomplished using labelled property graphs constructed via Cypher queries[1]. Unlike formal knowledge graph methods relying on RML/R2RML mappings [21], these mappings remain implicit in query structures, limiting transparency and reuse. Later refinements [11] introduce JSON-based "semantic headers" for mapping, but this weak encoding raises concerns for logical consistency. EKGs also assume perfect data quality, reinforcing the need for formalized data quality handling within process mining knowledge representations.

DECLARE [8] is a declarative process modeling approach that specifies constraints (e.g., "A happens at most once," "C always follows B") using linear temporal logic (LTL) rather than imperative sequences like $A \implies B \implies C$, with a defined constraint vocabulary and diagrammatic representations.

DECLARE's use of LTL has key semantic limitations, as it models only sequential relationships within a single case and treats all constraint operands as homogeneous event entities. These limitations prevent explicit reconciliation of semantic heterogeneity issues, and leave no room to model important heterogeneous process relationships like participation or event-state relationships. Additionally, semantic integration is virtually impossible as DECLARE's point-based semantics cannot be mapped to interval-based semantics, to name one issue.

Data-aware DECLARE [9] extends DECLARE through an expanded metric first-order temporal logic (MFOTL) to incorporate data conditions via "event payloads", enhancing expressiveness but still reducing complex entities to homogeneous attribute-value pairs. This limitation, along with DECLARE's implicit process ontology, prevents fundamental inference and consistency checks, such as ensuring activities do not end before they begin. Other similar data-aware approaches are summarized in [10], where research in database theory dynamics for process analysis is presented. These approaches similarly use temporal logic and automated planning (AP) tools. However, our ontological critiques similarly hold, as the use of domain-specific schemata to validate logs fails to formally model fundamental ontological distinctions of processes. The reduction of diverse process and domain entities into homogeneous attribute-value pairs results in an inflexible and opaque semantics.

Ontology-Based Data Access (OBDA) for Process Mining [22], notably represented by onprom [15], maps data to an RDF graph to enable SPARQL queries for

[1] `neo4j.com/docs/cypher-manual/current/introduction/`

entity-centric retrieval. However, it suffers from poor expressiveness and neglects data pre-processing and quality issues. RDF graphs in OBDA support only subsumption, disjunction, and cardinality, which limits its ability to represent complex process concepts like temporal relationships. Moreover, OBDA relies on annotations and ad-hoc reasoning, limiting its formal reasoning capabilities.

Semantic Web Ontologies have been applied in process mining, with relatively high expressiveness, but still facing key limitations. For example, the Event Ontology (EVO) [14] serves as a taxonomy for event types like start or monitoring events, while BPMO [13] can annotate events according to control flow structures of common process models. Despite using OWL ontologies, these approaches rely on simple subsumption and domain/range axioms, making them more suitable for semantic tagging than for robust reasoning in diverse process mining scenarios. Other semantic web ontologies do not directly cover process mining but focus on events. These ontologies, however, are similarly limited by poor expressiveness, as the findings of Katsumi and Gruninger [23] indicates. These findings demonstrate that several prominent event ontologies are unable to handle basic competency questions of processes. Similarly, Benevides et al. [24] found that semantic web ontologies are unsuitable for representing foundational ontologies of process in their attempt to adapt the expressive UFO-B process ontology into a semantic web format.

Upper Ontologies model general, domain-independent knowledge that inevitably includes processes. Notable upper ontologies in this category include DOLCE [5] and UFO [6]. Being expressive and domain-independent ontologies, they consequentially carry conceptual bloat and redundancy of knowledge for process mining. For instance, to model the single simple event of a man running, DOLCE requires relevant perdurants, accomplishments, agentive physical objects, temporal qualities, and temporal regions. These ontologies' attempt to provide a general ontological reality leads to a confusing data structure for specific grounded applications like process mining. Additionally, these upper ontologies lack any tooling, methodology, or guidance for integration with domain datasets.

The Process Specification Language (PSL) [7] is a first-order logic ontology designed for the exchange of process knowledge among disparate information systems. While it is recognized as an ISO standard (ISO 18629 [1]), its application for event logs is unclear due to their inherent semantic heterogeneity. PSL also lacks a means of integration with domain data or a practical way to handle exceptional behaviours like activity attempts, compounded by (general) undecidability issues in first-order logic reasoning.

4 The Process Meaning Pattern Framework

The process meaning pattern (PMP) framework formalizes the ontological commitments of process mining through semi-automated integration of ontology reasoning with data analysis tooling. In this section, we present the architecture of the frame-

[1] `https://www.iso.org/standard/35431.html`

work and formally specify key demonstrative patterns with first-order logic and a logic programming implementation [1].

4.1 Overview

The high-level architecture of PMP (figure 4.1) can be viewed through two layers: a **knowledge layer** and an **implementation layer**, representing the formally encoded components of the framework, and the supporting software to integrate those formal components, respectively.

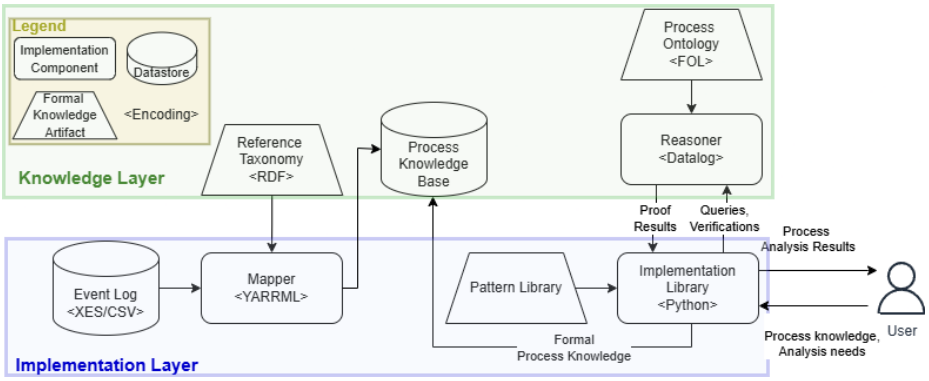

Fig. 1. Basic System Architecture for the PMP Framework

The end-user implementation is designed to mimic well-established libraries like pandas [2], with ontology-interpreted data structures rather than dataframes. Raw event log data is mapped into a set of simple RDF triples (e.g., `hasRecordedTime(bakePizza, 12/03/24 10:15)`, `case(baking03)`), via a lightweight reference taxonomy (encoded via a YARRML file [25]), forming the initial process knowledge base.

The central construct of our framework is the **process meaning pattern**: specification of process statements, queries, and checks, like "What kinds of activities cause the status X to change?" or "X is a subactivity of Y". The patterns are specified via first-order logic sentence patterns in the language of the process ontology. While this is structurally similar to approaches like DECLARE [8], it has a richer semantics due to its reliance on an expressive process ontology. For the purpose of demonstrating the framework in this paper, we use a process ontology based on the process specification language (PSL) [7], though its aforementioned challenges will need to be addressed for a full implementation of the framework. By iteratively validating and extending the knowledge base through these patterns, PMP enables practitioners to formally capture assumptions, refine their analyses, and integrate provenance information directly into the process mining pipeline, rather than apply it in an ad-hoc manner.

Process Meaning patterns can be broadly categorized into three types: descriptors, facts, and queries; each of these is defined in table 1.

[1] The full implementation of the framework is available at `github.com/riley-momo/ProcessMeaningPatternsPython`

[2] https://pandas.pydata.org/

Table 1. Summary of Process Meaning Pattern Categories

Category	Description	Example
Descriptor	Specifies process behaviour using domain-specific vocabulary, quantified over ontology concepts	Gold members receive a 10% discount at checkout
Fact	States simple, quantifier-free knowledge about a process using domain-specific terms	Bob's loan was approved
Query	Validates or infers behavior by checking entailment of quantified statements over ontology concepts	What are all the instances of hand-offs in loan approvals?

Consider an exemplary instance of a descriptor pattern: "When a fragile object is dropped, it breaks." - this can be expressed with the following first-order sentence in the language of the ontology:

$$(\forall o)\, occurrence\_of(o, drop) \wedge prior(fragile, o) \implies holds(broken, o)$$

This sentence is a specific example of a **state-based effect (SBE)** pattern, since it specifies how a state effects the result of an activity occurring. This process meaning pattern can be expressed through the meta-logical definition:

$$SBE(a, f_1, f_2) := (\forall o)\, occurrence\_of(o, a) \wedge prior(f_1, o) \implies holds(f_2, o)$$

In natural language, this states: "If an action a occurs and a state f_1 holds prior, then another state f_2 must hold afterwards."

4.2 Reasoning with Patterns

In addition to PMP enabling formal representations of process knowledge, use of a reasoner enables inference of additional process facts, retrieval of knowledge, and verification of process data. The positioning of the process ontology as a core element from which patterns are specified and data is mapped results in more focused coverage than other upper ontologies [5,6], while also distinguishing domain-independent knowledge from domain process knowledge, unlike other declarative approaches [8,9]. Additionally, reasoning through patterns presents an explicit connection between domain knowledge and process mining scenarios. For instance, the query pattern `handOffs(p)` is a unary query pattern, accepting a singular process variable, indicating this pattern may be instantiated without defining any process description or domain-level information. Conversely, a descriptor pattern like `stateBasedEffect(a, f1, f2)` accepts several arguments, indicating the knowledge necessary to express this rule. Since reasoning is framed in terms of the process ontology and a structured vocabulary of patterns, its complexity can be managed by restricting the expressiveness of allowed sentence patterns. This enables some control over decidability and tractability in process mining applications.

4.3 Implementation and Demonstration

To demonstrate our framework in action, we draw from a real-world case study from Volvo IT Belgium [1] with reference to specific python and datalog code.

[1] `ais.win.tue.nl/bpi/2013/challenge.html`

One of the key challenges for Volvo IT Belgium (and for large enterprises in general) is understanding organizational dynamics within their processes. In this case, identifying teams and cases involved in "ping-pong" (also called multi-hop) behaviour, where several hand-offs occur while handling an IT incident. By applying PMP, we address this scenario by identifying and inferring instances of ping-pong behaviour directly from the dataset, demonstrating the power of our approach in capturing and reasoning about process dynamics. This demonstration will be presented in terms of the practitioners workflow.

Firstly, the event log data (modelled in CSV format) will be mapped into a process knowledge base containing a record of all the events pertaining to the IT incident management process. In this case, the mapping outputs to a datalog format, but our implementation also currently supports RDF, prover9, and CLIF formats.

Next, the pattern relevant to the current analysis needs would be selected, the ping-pong pattern in this case. In a future implementation, selection from a library of patterns would be integrated into the python library, but for now it must be manually entered in the datalog file. The ping-pong behaviour is defined by the following first-order sentence in the language of the ontology:

$$\forall(c)\, \mathit{ping\_pong}(c) \iff \exists(e_1,e_2,e_3,e_4,r_1,r_2)\, \mathit{hand\_off}(e_1,r_1,e_2,r_2,c) \wedge$$
$$\mathit{hand\_off}(e_3,r_3,e_4,r_4,c) \wedge (e_1 \neq e_3) \wedge (e_2 \neq e_4)$$

The `ping_pong(c)` pattern is true of a case `c`, and is defined in terms of the sub-pattern `hand_off(e1, r1, e2, r2, c)`, which denotes a hand-off between the subsequent events `e1` and `e2` from resource `r1` to resource `r2` in the case `c`. Notice that this definition fits the query pattern as an existential conjunction of literals, and that ping pong is defined only in terms of the provided data; it does not require any background or domain-specific information to be applied.

When the query pattern is run, the reasoner will firstly check consistency of the pattern instance and knowledge base, and then return satisfying instances of the query, if any exist [1]. In this example, the datalog query pattern being matched is:

```
ping_pong(C) :-
    hand_off(R1, R2, E1, E2, C),
    hand_off(R3, R4, E3, E4, C),
    \+ (E1 = E3),
    \+ (E2 = E4).
```

The reasoner's output returns all cases with matching ping-pong behaviour, and will enumerate all proofs of this; this is equivalent to identifying all the possible pairs of hand-offs in each satisfying case. Given the proof output of the reasoner, practitioners are able to analyze each specific instance contributing to the ping-pong behaviour and examine them more closely.

Going beyond the ping-pong example, the intuitive and parametric definition of patterns enables complex queries to easily be expressed to better understand the process dynamics between resources. For example, "Which hand-off events involve the finance department taking over"? can simply be expressed through the query: $\mathit{hand\_off}(e, r_1, e_2,$

[1] In its current state datalog reasoning is semi-automated but future work will directly integrate the reasoner into the python environment for greater ease of use.

$Finance, c$). Additionally, given the positioning of this reasoning as part of a pre-defined vocabulary of patterns, as well as reasoning being carried out in datalog, it results in many computational benefits. Firstly, given that any pattern being checked will be evaluated through a datalog expression, it will be equivalent to a horn clause and thus be decidable. Secondly, it opens up new avenues for non-monotonic reasoning such as defaults or exceptions.

5 Conclusion and Future Work

The process meaning patterns framework advances ontology-aware process mining by bridging practical process mining analysis with formal ontology. Addressing challenges such as semantic heterogeneity, data quality, and reasoning tractability, it enhances the transparency, interpretability, and verification of process mining analyses. Through formal axiomitization, pragmatic implementation, and real-world applications, it demonstrates both theoretical rigour and practical utility.

Future work includes expanding the pattern library, refining the implementation including direct integration of datalog reasoning and pattern selection with python, and applying the framework as a benchmark of other process ontologies' practical utility. By grounding process mining in a structured yet practical framework, this work supports more consistent, reusable, and insightful analyses, benefiting both research and industry.

References

1. Wil Van Der Aalst, Arya Adriansyah, Ana Karla Alves De Medeiros, Franco Arcieri, Thomas Baier, Tobias Blickle, Jagadeesh Chandra Bose, Peter Van Den Brand, Ronald Brandtjen, Joos Buijs, et al. Process mining manifesto. In *Business Process Management Workshops: BPM 2011 International Workshops, Clermont-Ferrand, France, August 29, 2011, Revised Selected Papers, Part I 9*, pages 169–194. Springer, 2012.
2. Wil MP van der Aalst. Process mining: a 360 degree overview. In *Process Mining Handbook*, pages 3–34. Springer, 2022.
3. Suriadi Suriadi, Robert Andrews, Arthur HM ter Hofstede, and Moe Thandar Wynn. Event log imperfection patterns for process mining: Towards a systematic approach to cleaning event logs. *Information systems*, 64:132–150, 2017.
4. Iezalde F Lopes and Diogo R Ferreira. A survey of process mining competitions: the bpi challenges 2011–2018. In *Business Process Management Workshops: BPM 2019 International Workshops, Vienna, Austria, September 1–6, 2019, Revised Selected Papers 17*, pages 263–274. Springer, 2019.
5. Stefano Borgo, Roberta Ferrario, Aldo Gangemi, Nicola Guarino, Claudio Masolo, Daniele Porello, Emilio M Sanfilippo, and Laure Vieu. Dolce: A descriptive ontology for linguistic and cognitive engineering. *Applied ontology*, 17(1):45–69, 2022.
6. Giancarlo Guizzardi, Alessander Botti Benevides, Claudenir M Fonseca, Daniele Porello, João Paulo A Almeida, and Tiago Prince Sales. Ufo: Unified foundational ontology. *Applied ontology*, 17(1):167–210, 2022.
7. Michael Grüninger. Ontology of the process specification language. In *Handbook on ontologies*, pages 575–592. Springer, 2004.
8. Claudio Di Ciccio, Marco Montali, et al. Declarative process specifications: Reasoning, discovery, monitoring. *Process mining handbook*, 448:108–152, 2022.

9. Giacomo Bergami, Fabrizio Maria Maggi, Andrea Marrella, and Marco Montali. Aligning data-aware declarative process models and event logs. In *Business Process Management: 19th International Conference, BPM 2021, Rome, Italy, September 06–10, 2021, Proceedings 19*, pages 235–251. Springer, 2021.
10. Diego Calvanese, Giuseppe De Giacomo, and Marco Montali. Foundations of data-aware process analysis: a database theory perspective. In *Proceedings of the 32nd ACM SIGMOD-SIGACT-SIGAI symposium on Principles of database systems*, pages 1–12, 2013.
11. Ava Swevels, Dirk Fahland, and Marco Montali. Implementing object-centric event data models in event knowledge graphs. In *International Conference on Process Mining*, pages 431–443. Springer, 2023.
12. Shahrzad Khayatbashi, Olaf Hartig, and Amin Jalali. Transforming event knowledge graph to object-centric event logs: A comparative study for multi-dimensional process analysis. In *International Conference on Conceptual Modeling*, pages 220–238. Springer, 2023.
13. Barry Norton, Liliana Cabral, and Jörg Nitzsche. Ontology-based translation of business process models. In *2009 Fourth International Conference on Internet and Web Applications and Services*, pages 481–486. IEEE, 2009.
14. Carlos Pedrinaci and John Domingue. Towards an ontology for process monitoring and mining. In *CEUR Workshop Proceedings*, volume 251, pages 76–87, 2007.
15. Diego Calvanese, Tahir Emre Kalayci, Marco Montali, and Stefano Tinella. Ontology-based data access for extracting event logs from legacy data: the onprom tool and methodology. In *Business Information Systems: 20th International Conference, BIS 2017, Poznan, Poland, June 28–30, 2017, Proceedings 20*, pages 220–236. Springer, 2017.
16. Zaharah A Bukhsh, Aaqib Saeed, and Remco M Dijkman. Processtransformer: Predictive business process monitoring with transformer network. *arXiv preprint arXiv:2104.00721*, 2021.
17. Sebastian Dunzer, Matthias Stierle, Martin Matzner, and Stephan Baier. Conformance checking: a state-of-the-art literature review. In *Proceedings of the 11th international conference on subject-oriented business process management*, pages 1–10, 2019.
18. Christian W Gunther and HMW Verbeek. Xes-standard definition. 2014.
19. Malte Thiede, Daniel Fuerstenau, and Ana Paula Bezerra Barquet. How is process mining technology used by organizations? a systematic literature review of empirical studies. *Business Process Management Journal*, 24(4):900–922, 2018.
20. Dirk Fahland. Process mining over multiple behavioral dimensions with event knowledge graphs. In *Process mining handbook*, pages 274–319. Springer, 2022.
21. Anastasia Dimou, Miel Vander Sande, Pieter Colpaert, Ruben Verborgh, Erik Mannens, and Rik Van de Walle. Rml: A generic language for integrated rdf mappings of heterogeneous data. *Ldow*, 1184, 2014.
22. Guohui Xiao, Diego Calvanese, Roman Kontchakov, Domenico Lembo, Antonella Poggi, Riccardo Rosati, and Michael Zakharyaschev. Ontology-based data access: A survey. International Joint Conferences on Artificial Intelligence, 2018.
23. Megan Katsumi and Michael Grüninger. Using psl to extend and evaluate event ontologies. In *International Symposium on Rules and Rule Markup Languages for the Semantic Web*, pages 225–240. Springer, 2015.
24. Alessander Botti Benevides, Jean-Rémi Bourguet, Giancarlo Guizzardi, Rafael Peñaloza, and João Paulo A Almeida. Representing a reference foundational ontology of events in sroiq. *Applied Ontology*, 14(3):293–334, 2019.
25. Pieter Heyvaert, Ben De Meester, Anastasia Dimou, and Ruben Verborgh. Declarative rules for linked data generation at your fingertips! In *The Semantic Web: ESWC 2018 Satellite Events: ESWC 2018 Satellite Events, Heraklion, Crete, Greece, June 3-7, 2018, Revised Selected Papers 15*, pages 213–217. Springer, 2018.