



Transactions on
Graph Data and Knowledge

Volume 2 | Issue 1 | May, 2024

Special Issue: Trends in Graph Data and Knowledge – Part 2

Edited by

Aidan Hogan
Ian Horrocks
Andreas Hotho
Lalana Kagal

ISSN 2942-7517

TGDK Special Issue Editors

Aidan Hogan 

DCC, Universidad de Chile, IMFD, Chile
ahogan@dcc.uchile.cl

Ian Horrocks 

University of Oxford, U.K.
ian.horrocks@cs.ox.ac.uk

Andreas Hotho 

Department of Informatics, University of Würzburg,
Germany
hotho@informatik.uni-wuerzburg.de

Lalana Kagal 

Massachusetts Institute of Technology, Cambridge,
MA, USA
lkagal@csail.mit.edu

ACM Classification 2012

Computing methodologies → Knowledge representation and reasoning; Information systems → Semantic web description languages; Information systems → Graph-based database models; Computing methodologies → Machine learning; Theory of computation → Graph algorithms analysis; Mathematics of computing → Graph theory

Published online and open access by

Schloss Dagstuhl – Leibniz-Zentrum für Informatik GmbH, Dagstuhl Publishing, Saarbrücken/Wadern, Germany.

Online available at

<https://www.dagstuhl.de/dagpub/2942-7517>.

Publication date

May, 2024

License

This work is licensed under a Creative Commons Attribution 4.0 International license (CC BY 4.0): <https://creativecommons.org/licenses/by/4.0>.



In brief, this license authorizes each and everybody to share (to copy, distribute and transmit) the work under the following conditions, without impairing or restricting the authors' moral rights:

- Attribution: The work must be attributed to its authors.

The copyright is retained by the corresponding authors.

Digital Object Identifier

10.4230/TGDK.2.1.0

Aims and Scope

Transactions on Graph Data and Knowledge (TGDK) is an Open Access journal that publishes original research articles and survey articles on graph-based abstractions for data and knowledge, and the techniques that such abstractions enable with respect to integration, querying, reasoning and learning. The scope of the journal thus intersects with areas such as Graph Algorithms, Graph Databases, Graph Representation Learning, Knowledge Graphs, Knowledge Representation, Linked Data and the Semantic Web. Also in-scope for the journal is research investigating graph-based abstractions of data and knowledge in the context of Data Integration, Data Science, Information Extraction, Information Retrieval, Machine Learning, Natural Language Processing, and the Web.

The journal is Open Access without fees for readers or for authors (also known as Diamond Open Access).

Editors in Chief

- Aidan Hogan
- Ian Horrocks
- Andreas Hotho
- Lalana Kagal

Editorial Office

Schloss Dagstuhl – Leibniz-Zentrum für Informatik
TGDK, Editorial Office
Oktavie-Allee, 66687 Wadern, Germany
tgdk@dagstuhl.de

<https://www.dagstuhl.de/tgdk>


■ Contents


Trends in Graph Data and Knowledge

Towards Representing Processes and Reasoning with Process Descriptions on the Web <i>Andreas Harth, Tobias Käfer, Anisa Rula, Jean-Paul Calbimonte, Eduard Kamburjan, and Martin Giese</i>	1:1–1:32
Grounding Stream Reasoning Research <i>Pieter Bonte, Jean-Paul Calbimonte, Daniel de Leng, Daniele Dell’Aglia, Emanuele Della Valle, Thomas Eiter, Federico Giannini, Fredrik Heintz, Konstantin Schekotihin, Danh Le-Phuoc, Alessandra Mileo, Patrik Schneider, Riccardo Tommasini, Jacopo Urbani, and Giacomo Ziffer</i>	2:1–2:47
Semantic Web: Past, Present, and Future <i>Ansgar Scherp, Gerd Groener, Petr Škoda, Katja Hose, and Maria-Esther Vidal</i>	3:1–3:37
Logics for Conceptual Data Modelling: A Review <i>Pablo R. Fillottrani and C. Maria Keet</i>	4:1–4:30
Standardizing Knowledge Engineering Practices with a Reference Architecture <i>Bradley P. Allen and Filip Ilievski</i>	5:1–5:23

■ List of Authors


Bradley P. Allen  (5)
University of Amsterdam, The Netherlands

Pieter Bonte  (2)
Department of Computer Science, KU Leuven
Campus Kulak, Belgium


Jean-Paul Calbimonte  (1, 2)
University of Applied Sciences and Arts Western
Switzerland HES-SO, Sierre, Switzerland; The
Sense Innovation and Research Center, Lausanne,
Switzerland


Daniel de Leng  (2)
Linköping University, Sweden

Daniele Dell’Aglia  (2)
Aalborg University, Denmark


Emanuele Della Valle  (2)
DEIB - Politecnico di Milano, Italy


Thomas Eiter  (2)
Technische Universität Wien, Austria

Pablo R. Fillottrani  (4)
Universidad Nacional del Sur, Bahía Blanca,
Argentina; Comisión de Investigaciones Científicas,
Provincia de Buenos Aires, Argentina


Federico Giannini  (2)
DEIB - Politecnico di Milano, Italy

Martin Giese  (1)
University of Oslo, Norway


Gerd Groener  (3)
Carl Zeiss SMT GmbH, Oberkochen, Germany

Andreas Harth  (1)
Friedrich-Alexander-Universität
Erlangen-Nürnberg, Germany; Fraunhofer Institute
for Integrated Circuits IIS, Nürnberg, Germany

Fredrik Heintz  (2)
Linköping University, Sweden

Katja Hose  (3)
TU Wien, Austria


Filip Ilievski  (5)
Vrije Universiteit, Amsterdam, The Netherlands

Eduard Kamburjan  (1)
University of Oslo, Norway


C. Maria Keet  (4)
Department of Computer Science, University of
Cape Town, South Africa

Tobias Käfer  (1)
Karlsruhe Institute of Technology (KIT), Germany


Danh Le-Phuoc  (2)
Technical University Berlin, Germany


Alessandra Mileo  (2)
Insight Centre for Data Analytics, Dublin City
University, Ireland


Anisa Rula  (1)
University of Brescia, Italy


Konstantin Schekotihin  (2)
Alpen-Adria-Universität Klagenfurt, Austria

Ansgar Scherp  (3)
Ulm University, Germany


Patrik Schneider  (2)
Technische Universität Wien, Austria; Siemens AG,
Chemnitz, Germany

Riccardo Tommasini  (2)
INSA Lyon, CNRS LIRIS, France; University of
Tartu, Estonia

Jacopo Urbani  (2)
Vrije Universiteit Amsterdam, The Netherlands

Maria-Esther Vidal  (3)
Leibniz University of Hannover, Germany;
TIB-Leibniz Information Centre for Science and
Technology, Hannover, Germany

Giacomo Ziffer  (2)
DEIB - Politecnico di Milano, Italy

Petr Škoda  (3)
Department of Software Engineering, Faculty of
Mathematics and Physics, Charles University,
Prague, Czech Republic

Towards Representing Processes and Reasoning with Process Descriptions on the Web

Andreas Harth  

Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany
Fraunhofer Institute for Integrated Circuits IIS, Nürnberg, Germany

Tobias Käfer  

Karlsruhe Institute of Technology (KIT), Germany

Anisa Rula  

University of Brescia, Italy

Jean-Paul Calbimonte  

University of Applied Sciences and Arts Western Switzerland HES-SO, Sierre, Switzerland
The Sense Innovation and Research Center, Lausanne, Switzerland

Eduard Kamburjan  

University of Oslo, Norway

Martin Giese  

University of Oslo, Norway

Abstract

We work towards a vocabulary to represent processes and temporal logic specifications as graph-structured data. Different fields use incompatible terminologies for describing essentially the same process-related concepts. In addition, processes can be represented from different perspectives and levels of abstraction: both state-centric and event-centric perspectives offer distinct insights into the underlying processes. In this work, we strive to unify the representation of processes and related concepts by leveraging the power of knowledge graphs. We survey approaches to representing processes and reasoning with process descriptions from different fields and provide a selection of scenarios to help

inform the scope of a unified representation of processes. We focus on processes that can be executed and observed via web interfaces. We propose to provide a representation designed to combine state-centric and event-centric perspectives while incorporating temporal querying and reasoning capabilities on temporal logic specifications. A standardised vocabulary and representation for processes and temporal specifications would contribute towards bridging the gap between the terminologies from different fields and fostering the broader application of methods involving temporal logics, such as formal verification and program synthesis.

2012 ACM Subject Classification Information systems → Semantic web description languages; Theory of computation → Program semantics; Applied computing → Business process modeling; Applied computing → Event-driven architectures; Computing methodologies → Temporal reasoning; Computing methodologies → Ontology engineering

Keywords and phrases Process modelling, Process ontology, Temporal logic, Web services

Digital Object Identifier 10.4230/TGDK.2.1.1

Category Survey

Funding Harth acknowledges support from the German Federal Ministry of Education and Research via grant MOSAIK (01IS18070A), Käfer from the German Research Foundation (DFG) via grant FOR 5339 (459291153), Rula from the M.U.R. Italian Minister of University within the PRIN 2022 Programme via grant NEXTCART (2022YXXZH5), Calbimonte from the Swiss National Science Foundation via the grant StreamKG (213369) and Kamburjan and Giese from the Research Council of Norway via grants PetWIN (294600) and SIRIUS (237898). The article is based upon work conducted as part of the COST Action Distributed Knowledge Graphs (CA19134) supported by COST (European Cooperation in Science and Technology).



© Andreas Harth, Tobias Käfer, Anisa Rula, Jean-Paul Calbimonte, Eduard Kamburjan, and Martin Giese; licensed under Creative Commons License CC-BY 4.0

Transactions on Graph Data and Knowledge, Vol. 2, Issue 1, Article No. 1, pp. 1:1–1:32



Transactions on Graph Data and Knowledge

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Acknowledgements We acknowledge the constructive comments of the anonymous reviewers. Harth acknowledges discussions over the years with Victor Charpenay, Stefan Decker, Dieter Fensel, Justus Fries, Stefan Jablonski and Daniel Schraudner regarding the formalisation of processes. Harth and Käfer acknowledge discussions with Brian Logan and the participants of the Dagstuhl Seminar 23081 Agents on the Web.

Received 2023-06-30 **Accepted** 2023-11-17 **Published** 2024-05-03

Editors Aidan Hogan, Ian Horrocks, Andreas Hotho, and Lalana Kagal

Special Issue Trends in Graph Data and Knowledge – Part 2

1 Introduction

Representing knowledge – describing what is true – is a mature field with a well-established terminology and existing standards. Knowledge representation with ontologies, using graph-structured representations, is standardised with W3C recommendations around the Resource Description Framework (RDF) [27], RDF Schema [17] and the Web Ontology Language OWL [11]. The languages specified by the W3C enjoy mature tool support, are widely applied in many different domains and have a clear connection with web architecture (Linked Data) to share and access data. The adoption of principles for sharing data in research, especially regarding findability, accessibility, interoperability and reusability (FAIR data [107]) provides organisational guidance. However, knowledge representation with ontologies has a focus on describing what is the case. Although OWL Time [26] provides primitives to represent time, support for representing changes and temporal aspects is often limited in the existing languages.

Representing processes – describing what is happening – is, in contrast, less standardised and there is less agreement on terminology and languages across fields. While many research fields are concerned with the temporal representation of processes and related concepts, each field has developed their own terminology and methods to address their requirements. Although many fields try to represent essentially the same concepts, terminology is often incompatible across approaches. Especially in the Semantic Web community, there is a lack of agreed-upon terminology and lack of actively-used vocabularies for representing processes and related concepts.

The literature on research around representing and reasoning with processes is vast and scattered across many fields. There is potential to bring hitherto unconnected languages, approaches and systems together and put sophisticated methods around model checking and planning into the hands of more users. Our aim is to try to capture the core of processes and process-related concepts, applicable to many different domains, by trying to distil essential aspects of events and processes.

We start with giving a brief historical overview of a selection of general works on events and processes from a selection of fields.

- **Philosophy of Language and Linguistics:** Publications from the field of philosophy of language and linguistics provide arguments for approaches to represent actions and events, but little in terms of mathematical formalisation. Linguistics and philosophy of language had notable publications starting in the 1950s and 1960s with a focus on understanding the nature of events. Vendler [105] investigates verb types and distinguishes temporal entities according to whether they occur gradually or instantaneously and whether they have an endpoint or not. States describe what is the case and last for a period of time (“A loved somebody from t_1 to t_2 ”), activities are ongoing and do not have an endpoint (“A was running at time t ”), accomplishments are incremental and have an endpoint (“A was drawing a circle at t ”), and achievements occur instantaneously and have an endpoint (“A won a race between t_1 and t_2 ”). Davidson investigates the structure of action sentences [28], assuming a universe that contains particular events. In a follow-up work, Davidson addresses a criticism regarding

recurrence of events, *i.e.*, that certain events can occur multiple times [29] and discusses the possibility of events as universals. More recently, Galton argues that processes are “patterns of occurrences” [43] and thus treats processes as a kind of universal. The works in the field of philosophy of language and linguistics provide only rudimentary material that would lead to a mathematically rigorous formalisation that supports deductive reasoning.

- **Temporal Logic and Formal Methods:** Many of today’s knowledge representation languages build on subsets or variants of first-order logic, often formalised using interpretations and models over a universe (or world). The introduction of modal logic brought ways to talk about necessity and possibility and truth relative to multiple worlds. The formalisation is based on Kripke structures (or transition systems). Prior developed Tense Logic [91] in the 1950s using modal operators to represent temporal aspects. Pnueli [90] introduced Linear-time Temporal Logic (LTL) in the 1970s and 1980s to represent and reason about the behaviour of computer programs. Computational Tree Logic (CTL) [40] is an alternative formalisation to represent and reason about the behaviour of (technical) systems. While LTL uses a linear time model, CTL uses a branching time model. Both temporal logics provide the basis for formally proving correctness of programs via model checking. While LTL and CTL typically operate over states expressed in propositional logic formulas, Abstract State Machines (ASMs) [53] (initially called Evolving Algebras) provide support for full first-order logic structures as state representation. ASMs have been used to formalise the semantics of programming languages, for instance. A related approach for formally treating the behaviour of systems is described by McDermott [76]. With the right descriptions of operations (with pre- and postconditions), a planner can synthesise a program (a sequence of operations) that reaches a given goal. Many of these approaches focus on representing the dynamics of systems as a progression of state. Other approaches focus on representing the behaviour and the interaction between processes. Communicating State Machines [16] formalise processes that can send and receive messages. Similarly, Communicating Sequential Processes [56] and Milner’s work on a Calculus of Communicating Systems [77] and the π -calculus [78] have a focus on communication and interaction between processes.
- **Business Processes and Workflows:** While approaches from the philosophy of language and linguistics and from temporal logics and formal methods provide theory, the field of business processes and workflows apply process descriptions and executions in a business context. The focus in modelling business processes is on what is happening (events) and not what is true (states). That is, in the business process community [96], processes are primarily modelled as collections of activities (diverging from Vendler’s definition of activity). Event-driven process chains [68] are an early approach to represent business processes. Often, Petri nets [88] are used to formalise the different business process and workflow languages. Process models (or workflows, *i.e.*, processes that have a unique source place and a unique end place [36]) use a “directly-follows” relation between activities to describe a process. MOBILE [58] provides a modular workflow model and architecture for implementing business processes in office environments and distinguishes between functional, behavioural, organisational and informational aspects. In the workflow field, researchers have identified patterns related to control flow and data (states) [96] to be able to catalogue the features of workflow languages and systems. More recently, approaches such as DECLARE [87] and Guard-Stage-Milestone (GSM) [57] break up the rather rigid description of traditional process models and encode relations and dependencies between temporal entities in a more flexible manner. Rather than just using the immediate-next relation to describe processes, these approaches allow for describing sets of “directly-follows” processes. Finally, in the last decades the field of process mining [103] has gained prominence, to re-discover the control flow from traces of process executions.

Next to the overview of general approaches from various fields, we survey the state of the art concerning representing processes and related concepts with ontological modelling in RDF in Section 2.

In this work we identify the main concepts that allow the formal representation of processes, which could later form the basis for a common vocabulary using Semantic Web standards. We try to be as general as possible, but focus our investigation on processes that can be executed and observed via a web interface. While our focus is on describing technical processes, we try to be generic enough to not explicitly exclude physical, chemical, biological or social processes. We believe that such a formalisation of processes can help going beyond description of physical and abstract entities, expanding towards the specification of changes, including temporal relationships and reasoning. To illustrate the different ways of how these process description concepts might be used, in Section 3 we provide a set of scenarios in which we highlight their respective main requirements, in terms of process modelling patterns as well as querying and reasoning tasks.

Our idea of process representation includes the definition of steps, events and other elements relative to process modelling, catering for both the state-centric and the event-centric perspective. We also identify the need to provide means for distinguishing between particulars (occurrences) and universals (events, states, processes) and keeping a trace of the events and states during the unfolding of a process. Moreover, we illustrate how the specification of temporal properties can be used to encode restrictions on the temporal order of process elements. We develop the core concepts required for a core process ontology in Section 4.

Our endeavour has the potential to address not only representation of events and processes but also operational and reasoning tasks, beyond previous attempts with mixed success, such as those related to Semantic Web Services [75, 94]. Thus, next to aligning terminology and vocabulary, the overall challenge is to bring the sophisticated methods from temporal logics and formal methods to the Semantic Web community. We could use process representations to query the graphs describing the processes (interface with systems around first-order logic, *e.g.*, a query processor or a reasoner), to query the temporal structure (interface with systems around temporal logic, *e.g.*, a model checker or a planner), or to generate and accept a sequence of events (interface with process engines or stream reasoners). Thus, we propose a research agenda focusing on the core properties needed to represent processes and related concepts and their semantics. We discuss the main challenges for representing processes on the web in Section 5.

To begin to structure the discussion of processes and related concepts, we have identified five dimensions, based on a combination of classifications found in the literature [96, 14]. While the work in [96] distinguishes control, data and resource, the approach in [14] distinguishes process, knowledge and trace. We believe that process/control and data/knowledge capture a similar notion, respectively. We also consider that [96] and [14] do not cover operational and architectural concerns. We want to distinguish between systems that operate offline (*e.g.*, on historical recordings of process executions) and those that operate online (*e.g.*, managing a set of currently running processes). See Table 1 for an overview.

The remainder of the paper is structured as follows. We start with a survey of the state of the art in Section 2. We continue with the presentation of scenarios in Section 3 and then introduce the assumptions and core concepts related to describing processes in Section 4. Next, we identify the main challenges for representing processes on the web in Section 5 before concluding in Section 6.

■ **Table 1** Dimensions for representing processes and related concepts.

Dimension	Description
Control	The dimension refers to the representation of the process control flow, <i>i.e.</i> the dependencies (temporal or otherwise) between different parts that make up a process. The most basic control flow construct is that of a sequence. Other constructs include branching conditions and repetitions. Note that control flow can apply to the level of particulars (<i>i.e.</i> , an occurrence of a process) as well as universals (<i>i.e.</i> , a process).
Data	The dimension refers to the representation data (<i>i.e.</i> , data objects, data items, knowledge) related to processes. The data related to the various steps could refer to the state or the lifecycle of the domain objects created or manipulated by the process at a given point in time. We assume the regular distinction between particulars (<i>i.e.</i> , instances) and universals (<i>i.e.</i> , classes) as in traditional knowledge representation languages, though not all approaches use an object-oriented paradigm for data.
Resource	The dimension refers to the resources that drive the progress of a process (<i>e.g.</i> , human workers, compute threads, organisation and roles). In the resource dimension, often the distinction between particulars and universals comes up again.
Trace	The dimension refers to the representation of “instances” of a process (<i>i.e.</i> , an occurrence of a process) and the associated history of evolution. A trace can contain both occurrences of events and states. A trace needs to connect the constituent elements, possibly via shared data objects with unique identifiers (‘case ids’).
Online	The dimension refers to the underlying system architecture. In particular we distinguish between online and offline processing. Information about processes and process executions can be available in batch for offline processing or can be available at runtime in a live system that makes control information available (<i>e.g.</i> , upcoming next, the process branches according to some conditions).

2 State of the art

Given that process representations are investigated in many fields, in the following we can only provide an overview of a selection of publications. Our litmus test for inclusion of a work in this section is that the work considers both processes *and* ontological modelling or data represented in RDF. Such works contain ideas, concepts and use-cases from fields such as Upper Ontologies, (Scientific) Workflow Provenance, Semantic Web Services, Stream and Event Processing, Formal Methods and Business Process Management.

We group the presentation of publications according to fields. We start each group with a description of the aim of the field and then present related works and contrast them with our proposed approach. For comprehensive surveys, see [93], [31] and [9]. We give an overview in Table 2.

2.1 Upper ontologies

Upper ontologies aim to describe general concepts in modelling such that domain ontologies built using these upper ontologies follow a principled approach and do not conflate (metaphysical) categories. Often, upper ontologies distinguish between continuants (endurants) and occurrents (perdurants). Continuants are “three-dimensional”, *i.e.*, exist in space only, while occurrents are “four-dimensional”, *i.e.*, can exist in space and time.

The notion of a process appears in domain-independent and upper ontologies. While in the Descriptive Ontology for Linguistic and Cognitive Engineering (DOLCE) [15] itself does not specifically consider processes, other works use DOLCE to talk about processes and plans, *e.g.*,

■ **Table 2** Ontological and RDF-based works around processes with ideas, concepts and use-cases from different fields. We give whether (+) or not (-) a work allows for the representation of **C**ontrol, **D**ata, **R**esource, **T**race or **O**nline dimension of processes. Note that a + does not imply that semantics are defined for the representations. For the BPM works, we give their foundation (e.g. standard, process calculus), if applicable.

Field	Work	C	D	R	T	O
Upper Ontologies	DOLCE [15]	-	-	-	-	-
	BFO [84]	-	-	-	-	-
	OntoUML/UFO-B [4]	+	-	+	-	-
	Schema.org [52]	+	-	-	-	-
(Scientific) Workflow Provenance	PROV-O [59]	-	-	-	+	-
	OPMW [47]	+	-	-	+	-
	ProvONE ⁶	+	+	-	+	-
	ProvONE+ [18]	+	+	-	+	-
	PWO [44]	+	-	-	+	-
	Taverna [101]	+	-	-	+	-
	Pegasus [50]	+	-	-	+	-
	Kepler [5]	+	-	-	+	-
Semantic Web Services	Agarwal et al. [1] (π -calculus)	+	+	+	-	-
	OWL-S/DAML-S [8]	+	+	+	-	-
	WSMO [55]	+	+	+	-	-
Stream and Event Processing	RDF Stream Processors [13, 89, 20]	-	+	-	+	+
	Stream Reasoners [12, 80, 106]	-	+	-	+	+
	Complex Event Processors [7, 6]	+	+	-	+	+
Formal Methods	Program Ontologies [10, 72, 30]	+	-	-	+	-
	Data Access Tools [73, 85, 98, 71]	-	+	-	-	+
	Transition Systems with KGs [66, 22, 38]	+	+	+	-	+
	Model Checkers [109]	+	+	+	+	-
	Runtime Enforcement [64, 22]	+	+	+	-	+
	Runtime Checking [66, 38]	+	+	+	-	+
Business Process Management	BPMN ontology [95] (BPMN)	+	+	+	-	-
	BPMO [19] (BPEL/BPMN)	+	+	-	-	-
	Bertoli et al. [14] (BPMN)	+	+	-	+	-
	Koschmider et al. [69] (Petri Nets)	+	-	-	-	-
	Gasevic et al. [48] (Petri Nets)	+	-	-	-	-
	Thomas et al. [99] (EPC)	+	-	+	+	-
	WiLD [61]	+	+	-	-	+
	GSM4LD [63] (GSM/CMMN)	+	+	-	-	+

[83]. The Basic Formal Ontology (BFO) [84] contains a term for Process¹, and [97] details how to represent continuants (covering space) and occurrents (covering space and time). In the Unified Foundational Ontology part B (UFO-B), or UFO's incarnation OntoUML, events are considered [4],

¹ http://purl.obolibrary.org/obo/BFO_0000015

where an event is something that has a beginning and an end in time and where other things (specifically durants) may participate. One could compare an event in UFO-B to an activity or process in the terminology of the other surveyed fields. Then, the participants in an event could be regarded as the resource dimension. In addition, the mereological (*i.e.*, part-of) and historical relations between events could be regarded as the control dimension, albeit with only very few language constructs.

In Schema.org [52], there are no terms to model processes. However, special kinds of processes can be modelled, such as a how-to description or a (cooking) recipe², which, at least in the JSON representation, can consider sequential flow. However, in the translation to RDF, the ordering gets lost, but there is an ongoing discussion to change that³.

2.2 (Scientific) workflow provenance

Works in Scientific Workflows and Provenance are mainly concerned with modelling entities, activities that transformed them and agents who carry out the activities. Those works were developed first for the recording of the activity of scientific labs that process samples, *e.g.*, in biology.

The Open Provenance Model for Workflows (OPMW) is an ontology to describe the workflow traces and their templates of scientific processes [45]. Part of the OPMW work is the Provenance and Plans ontology [46] (P-Plan⁴), which extends the Provenance Ontology [59] (PROV-O⁵) with terminology to describe the plans (that is, the control flow) that govern the execution of scientific processes. P-Plan can be used to describe plans and establish a relationship between such plans and the provenance records in PROV-O that provide details with a focus on the entities and steps of past executions of scientific workflows. However, P-Plan does not capture elaborate control-flow constructs that are used in other types of workflows. A more recent approach is *ProvONE*⁶ for the publication of scientific workflows [47]. *ProvONE* provides constructs to model workflow specifications and workflow execution provenance. The OPMW and *ProvONE* capture traces of workflow executions. They aim to provide detailed records of how each step is performed, which aligns with the trace dimension. *ProvONE+* is an extension of *ProvONE* that specifically addresses the requirements for capturing the provenance of control-flow-driven workflows [18], including sequence, parallel split and synchronisation in the control category. *ProONE* and *ProvONE+*, address aspects of the data dimension in terms of how data is managed and utilised in control-flow-driven workflows. Outside the domain of scientific lab work, the Publishing Workflow Ontology (*PWO*) is an ontology for the description of workflows in scientific publishing [44]. They use ontology design patterns to address modelling requirements of workflows. *PWO* includes sequence, parallel split, synchronisation, exclusive choice and simple merge.

Various systems have been proposed to provide the environment for specifying and enacting workflows such as Taverna and Kepler but each of them partially provides all the patterns in the control-flow category. Taverna [101] uses SCUFL (the simple conceptual unified flow language, a data-flow language) to specify control constraints for execution ordering and conditional constructs, which can encompass various control flow patterns such as if/else or case structures. Kepler [5] provides support for branching and synchronisation. Wings for Pegasus [50] is a workflow system that uses semantic representations to describe the constraints of the data and computational

² <http://schema.org/Recipe>

³ <https://github.com/schemaorg/schemaorg/issues/1910>

⁴ <http://purl.org/net/p-plan>

⁵ <https://www.w3.org/TR/prov-o/>

⁶ <http://purl.org/provone>

steps in the workflow. These works aim to fulfil various requirements related to representing workflows, capturing provenance information, describing plans and steps, creating workflow templates, accommodating domain-specific extensions and managing control flow. In provenance execution, the focus is on capturing the detailed record of the actual execution steps. However, some of these models have limitations in accurately and fully specifying control-flow patterns for scientific workflows. This limitation primarily impacts the prospective provenance, which includes the structure and static context of a workflow. However, their primary focus is not on representing data or instances of processes but on control and provenance aspects. Moreover, they do not consider the representation of resources like human resources, computational resources, or roles in the workflow.

2.3 Semantic web services

Works in Semantic Web Services aim to annotate Web Services in order to facilitate automated discovery, composition (*e.g.*, into a process), invocation and re-use of such services. Their work builds on the works in Web Services that typically come without such annotations.

When business processes (*e.g.*, BPMN) are used to describe choreographies and orchestrations of electronic business activities, even beyond organisational boundaries [79], their execution requires interoperability between the electronic interfaces. Interoperability standards emerged from these needs, which in the context of process integration led to the creation of Web Services. Web Services⁷ allow for invoking operations tunnelled through HTTP POST requests. There is an entire family of standards, called ‘WS-*’, in the area of Web Services, which allow for, *e.g.*, service description or service discovery. To allow for the composition of Web Services, compositions via an orchestrator can be achieved using process languages such as BPEL⁸ (also known as BPEL4WS or WS-BPEL). [79] discusses that the life cycle of process instances and data items can be regarded as state machines.

Beyond these functionalities, Semantic Web Services aim at describing services through machine understandable representations, allowing the automated discovery, composition, invocation and reuse of services on the Web. Different alternatives have been proposed for representing Semantic Web services most prominently: OWL-S (Web Ontology Language for Web Services)⁹ and WSMO (Web Service Modelling Ontology)¹⁰. OWL-S (previously called DAML-S [8] – DARPA (Defense Advanced Research Projects Agency) Agent Markup Language for Services – consist of three main parts: an ontology describing what a service does; an ontology to describe the service process and an ontology to specify how to interact with the service. OWL-S contains terminology to describe (hierarchical) process models with the following control flow elements: Sequence, Split, Split-Join, Any-Order, Choice. The ‘service grounding’ aspect of OWL-S could be argued to hint at the resource aspect of processes. The input, output, precondition, effect of OWL-S’s ‘service profiles’ loosely relate to the data aspect. WSMO [94] proposes a conceptual model for describing Web services, separating the goals or expectations of service requesters from the actual descriptions of the service features, capabilities and interfaces. WSMO also introduces the concept of mediators allowing the conciliation of potential discrepancies between services. Execution in the context of WSMO, WSMX (Web Service Execution Environment) [55], is event-based. The ‘interface’ aspect of WSMO could be argued to hint at the resource aspect of processes. The input, output, precondition, effect of WSMO’s ‘capability’ loosely relate to the data aspect. Another approach

⁷ <https://www.w3.org/2002/ws/>

⁸ <http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html>

⁹ <http://www.w3.org/Submission/OWL-S/>

¹⁰ <http://www.w3.org/Submission/WSMO/>

from Semantic Web Services [1] contains a process language with formal semantics based on the π calculus. This language can express sequential, parallel (‘composition’) and conditional (‘summation’) control flow. The approach can model agents and services (corresponding to the resource aspect) next to involved ontologies (corresponding to the data aspect).

2.4 Stream and event processing

Stream processing addresses the problem of efficient managing and querying rapidly changing flows of data and events, typically augmented with time annotations. Hereby, time is a fundamental element for representing concepts such as order, simultaneity or concurrence of a given process. Systems, processing frameworks and query languages have been developed over the years, targeting the increasing demands of high-velocity data consumers, in multiple domains like healthcare monitoring, security surveillance, environmental sensing and the industrial Internet of Things.

Nevertheless, traditional stream processing often faces the challenge of handling heterogeneous data sources and thus different models, which are hard to integrate and reason upon. RDF stream processing, stream reasoning and complex event processors have been presented in the literature, addressing this challenge from different perspectives [34].

RDF stream modelling has been presented as an approach to extend RDF with explicit temporal annotations, in order to enable continuous query processing over events or stream items [34]. Following a graph-based representation with explicit semantics provides a richer modelling approach for data streams, also offering ways of overcoming semantic heterogeneity through streaming data integration, *e.g.*, through methods like ontology-based data access [21]. Existing modelling approaches like RDF Stream Processing in SPARQL (RSP-QL) [33], use point-in-time or interval semantics to represent time, defined at either the triple or graph level. Compatible implementations of continuous query processors for RDF streams include Continuous SPARQL (C-SPARQL) [13], Continuous Query Evaluation over Linked Streams (CQELS) [89], or SPARQLStream [20]. Processes in these RDF stream systems are often represented as states that may change over time (*e.g.*, sensor observations denoting the state of a monitored subject) and the queries are used to formalise specific types of instances or patterns – thus reflecting the control flow. A first attempt to formalise the description of RDF stream resources and endpoints is the Vocabulary and Catalog of Linked Streams (VoCaLS) [100], although it does not explicitly include the specification of streaming processes.

Stream reasoning has taken a step beyond continuous queries, exploring the possibility of exploiting the semantic relationships among events through reasoning tasks [34]. This sub-field has studied different approaches, many of which implement variations of incremental reasoning. View maintenance-inspired approaches include systems like the system by Volz et al. [106], which is based on the delete and re-derive (DReD) algorithm and in which axioms are added and removed according to the entailment rules. In these stream reasoners, processes and activities are not explicitly modelled, but such reasoners allow for reasoning tasks that involve the temporal structure. Such tasks typically allow for deriving new knowledge (*e.g.*, new events) as a result stream, thus generating new occurrences. Benchmarking and evaluation of different stream reasoning approaches is still an area under active exploration [51], given the heterogeneity of existing reasoning approaches and underlying temporal logics.

Semantic complex event processing (CEP) has studied the modelling and querying of streaming events, allowing the continuous querying of complex patterns. These patterns include, for example, finding sequences of events (*i.e.*, when an event e_1 is followed by another event e_2); or two evaluate if two events happen simultaneously [7]. CEP query languages like Event Processing SPARQL (EP-SPARQL) [6] allow using RDF triples to represent these events and add query language extensions that allow to express these temporal order and simultaneousness patterns,

which represent the control flow among events. The language includes the SEQ operator to find sequences of triple patterns according to a time-based order. Similarly, the simultaneous presence of triple patterns can be detected using the EQUALS operator. Later on, these CEP operators were formalised and incorporated into the RSEP-QL model that combines both CEP and RSP query models [32]. In these CEP implementations, the activities and processes are reflected in the queries, where the expected order of sequences, patterns and other conditions are specified. While this is convenient for monitoring tasks and summarisation, the definition of explicitly defined processes could potentially facilitate and optimise the execution of CEP and reasoning query patterns in streaming environments.

2.5 Formal methods

Formal methods are formal techniques to model and analyse the behaviour of systems. Those techniques are concerned with modelling a general notion of process, *i.e.*, describing any sequence of state changes and the nature of these state changes. They can be applied to system designs or specifications as they can be found in engineered systems, *e.g.*, railway systems [42]. They can also be applied to more abstract systems, such as programs, where they form the basis to describe both program semantics [108] and program correctness [54], or indeed any kind of system that inhibits behaviour, including natural and sociological ones. As such, formal methods, in particular those targeting programs and program development, are closely related to process modelling, because both are concerned with describing possible and allowed sequences of states and events. Therefore, we believe that the rich toolkit of languages and tools from formal methods can successfully be applied in process modelling.

As we are eventually looking for a formal grounding for a core process ontology, which shall allow for reasoning about control flow and data as prescribed by a process, we also summarise related work that connects ontologies with formal methods to describe programs.

Programs are similarly concerned with describing control flow and data, but even though programs focus on *generating* traces, the ontologies describing programs and traces formalise similar ideas as process ontologies and can be a valuable source of inspiration, in particular for domain-aware simulation.

We subdivide our discussion of ontologies and programs into three groups, depending on their task: ontologies that describe programs and their output, ontologies that enable data access from a running program and approaches that tightly couple programs and ontologies that interact during program execution.

2.5.1 Describing programs and software systems

The following approaches are concerned with the syntax of a program, or the processes and the final traces generated by the program – during execution these systems are not applicable. On a purely syntactic level, Atzeni and Atzori [10] propose an ontology to describe the source code of Java programs, but are not concerned with the runtime semantics, *i.e.*, the generated traces.

On a higher level of abstraction, Diepenbrock et al. [35] focus on describing the architecture of a microservices system, while Oberle et al. [81, 82] give an ontology for general concepts in software systems, aiming for, among others, conceptual clarity for middleware. Lee et al. [72] elaborate an ontology of concepts in Java to help introduce them in courses to novice programmers, by connecting them to concepts from education. Similarly, de Aguiar [30] have produced an ontology for concepts in object-oriented programming. Such systems are not targeting to describe processes, but the concepts in the language and methodology in which the programs/processes are implemented.

As for the output of programs, the BOLD tool of Pattipati et al. [86] and Al Haider et al. [2] use ontologies to describe *execution traces*, *i.e.*, logs, of programs with the aim to simplify access to these quickly growing structures for static analysis [110]. In a more specific setting, Din et al. [37] describe a simulator for the geological process, where the final output trace is exported as a knowledge graph for easy query access by domain experts. If the program is the description of the process in question, or a part of the program, *e.g.*, a method, then the output is a description of the generated traces. The approach is, however, not suitable for the kind of modelling we envision, as it heavily tends towards low-level implementation details.

2.5.2 Enabling safe data access

To ensure safe interactions between ontologies and programs, one approach is to ensure that data loaded by some query language adheres to the type system of the program. These approaches focus exclusively on the relation of occurrence and data. This has been implemented to express loadable data by using SHACL (Shapes Constraint Language) shapes [73] or SPARQL query containment [65] as static approaches at compile time, or by generating classes in the programming language from OWL ontologies, see, *e.g.*, [85, 98, 71].

These approaches focus on RDF data, which may contain the description or parameters of a process serialised in it, but again focus on low-level implementation details, specifically the class system of, mostly, object-oriented languages.

2.5.3 Making knowledge available at runtime

The final task we examine is to make knowledge available at runtime. These works are, thus, concerned with processes (as procedures, etc.), occurrences (as runtime objects) data (as data types) and resources (as environment). While there are several languages that are concerned with connecting some kind of knowledge with programs, we focus on those that are designed to operate either on description logics models or on RDF graphs, or at least have an extension that does so. One such language is Golog [74] and its concurrent extension ConGolog [49], that uses first-order logic guards to examine and pick elements from its own state. Zarriess et al. [109] extend ConGolog to integrate description logic into a concurrent extension of Golog. They also investigate model checking against CTL (Computation Tree Logic). Model checking does not execute the program and, thus, has no occurrences. ConGolog is minimalistic and, thus, more suited for high-level modelling of workflows than low-level approaches.

A more direct connection between knowledge and transition is established by language where the knowledge, *e.g.*, as a knowledge graph, is external to the program state, but can be queried and manipulated using special operators. Fagin et al. [41] use epistemic operators to explicitly distinguish between what an agent believes and what is the case in the world, leading to the concept of *knowledge-based* programs. An extension, again using description logics to model knowledge, is presented by Calvanese et al. [22], which instead of learning focuses on revision [67]: actions change the (global) knowledge, which must then be revisited and repaired to remain consistent. The approach is interesting for modelling the behaviour of ‘agents’ and is naturally concurrent, thus being another possible basis for modelling workflows in collaborative or competitive settings. This is akin to runtime enforcement, where an additional component fixes the trace during execution to ensure its some property, which Calvanese et al. understand as logical consistency.

The most direct connection is realised by semantically lifted programs [66], which interpret the runtime state as a knowledge graph using an external ontology and allow the program to access this knowledge graph at runtime. Semantically lifted programs support a modelling approach where knowledge and program data are unified, and a process is ontological and computational at

the same time, depending on how it is accessed. Ontology-mediated programs [38] work similarly and interpret parts of the knowledge graph as program variables, but the user must provide the mapping from program variables themselves. Furthermore, their approach requires the number of instances to be known upfront. Both approaches allow for checking conditions at runtime to realise (internalised) runtime monitoring.

Finally, semantic lifting can also be used to enforce that a program adheres to domain knowledge by a universal guard to check that a transition will result in a consistent, lifted knowledge graph [64]. This is implementing runtime enforcement on a trace level.

2.6 Business process management

The field of Business Process Management is concerned with the modelling of processes in businesses with the aims of this modelling ranging from communicating how (*i.e.*, according to which process) things should get done in a business to automating the process using information systems.

Business Process Model and Notation (BPMN) is a visual language to represent processes¹¹. The BPMN ontology provides a formal representation of the visual concepts defined in BPMN, such as steps, events, gateways and sequence flows [95]. The ontology in [95], as a representation of BPMN, covers the corresponding wide range of control flow patterns and allows for the specification of data requirements, such as input and output data and the assignment of roles or agents responsible for specific steps. The aim of the ontology was to faithfully represent the constraints of the visual modelling language, thus the necessary semantics for workflow execution is out of scope. The BPMN ontology has only limited support for modelling organisational or resource-related aspects of business processes.

The business process modelling ontology (BPMO) [19] has been developed to represent high-level business process workflow models, abstracting from existing notations and languages such as BPMN and BPEL (short for Business Process Execution Language¹²). BPMO focuses on the representation of process specification and includes various workflow elements such as tasks, events, block patterns (inspired by BPEL) and graph patterns (inspired by BPMN 1.0). These block and graph patterns serve as control flows, representing decision points within the workflow.

The work in [14] builds on [95] and also allows for modelling the data coming from process executions to enable querying and reasoning over the representations. The representations contain especially traces of past process executions next to the involved resources and data items. The authors use OWL reasoning (RDFS and OWL 2 RL) or SPARQL queries (with entailment regimes) to query processes, data and traces. The inferencing they consider is concerned with the resources that have been involved in specific traces.

Other works study the representation of Petri Nets using semantic technologies [69, 48]. Petri Nets are directed graphs consisting of *places* and *transitions* representing states and events/activities respectively. These works aim to enable the sharing of Petri nets in RDF or OWL. While the foundations of Petri Nets representations allow capturing different workflow patterns (*e.g.*, AND-Split, OR-Split, Loops, etc.), they are limited in their expressivity to the basic Petri Net language elements lacking consideration for extensions.

Another graphical modelling notation for business processes is the so-called event-driven process chain (EPC). EPC can be used to represent process models. Core language elements in EPC are functions (activities), events, control flow transitions, logical connectors (OR, AND, XOR) and resources. The authors of [99] investigate the semantic annotation of EPC process models.

¹¹ <http://www.omg.org/spec/BPMN/2.0/>

¹² <http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html>

Two approaches, WiLD (Workflows in Linked Data) and GSM4LD (Guard-Stage-Milestone for Linked Data), are designed to describe and execute workflows in Linked Data environments. WiLD [61] is a hierarchical process model [104], which allows for specifying sequential, parallel and conditional control flow. WiLD can also model process instances and together with its operational semantics in ASM4LD [60], workflows in WiLD can be executed. Correspondingly, the approach needs a notion of the data on which conditions are evaluated. A similar approach is taken in GSM4LD [63], which brings the Guard-Stage-Milestone (GSM) approach [57] (later standardised as CMMN – Case Management Model and Notation¹³) to Linked Data. Control in GSM is entirely determined data-driven, that is, via conditions evaluated on the environment and process-relevant milestones. Again, the operational semantics are given in ASM4LD. In contrast to GSM4LD, an execution of a WiLD workflow can give a light-weight process trace, by assigning finishing timestamps to finished activities, without additional modelling effort in the operational semantics. GSM4LD however supports cycles, thus to produce a complete trace would need extra data structures.

3 Scenarios

In this section, we provide an overview of various application areas that use processes and review the requirements outlined in these applications to capture various process aspects. These scenarios collectively cover a wide range of aspects, including control, data, resources, traces and architecture; the main task over these representations often relates to querying and reasoning over the representation of processes and related concepts. Additionally, we examine the features required of a process language, categorised based on *workflow patterns* [102] and we use WCP_n to indicate patterns related to control and WDP_m to indicate patterns related to data.

An aspect that was not explicitly mentioned in other works is that of hierarchies defined between processes. Concerning sub-process support the language should provide capabilities for defining a new relationship where the steps of a process include steps from other processes. We use H to indicate the hierarchical modelling pattern. These process hierarchies can be further divided into i) include, which indicates that a procedure is included in another procedure by specifying that the activities within a procedure are *inclusive* of the activities of another procedure. ii) extends, which identifies that a procedure presents a list of activities that are a variation of the activities of another procedure. iii) generalise, which is used in hierarchical process modelling to represent inheritance or specialisation relationships between processes.

3.1 Manufacturing process documentation

The *Manufacturing process documentation* use case involves the maintenance and adjustment of gear head lathe machinery¹⁴ as described in a maintenance manual. The process consists of a sequence of steps that must be completed in a sequential order to address maintenance tasks and ensure optimal machine performance. It is crucial to record the order in which these steps are carried out. The initial step in the sequence involves a safety check of the machinery. Upon completion of the preceding step, the process enables the following steps, including instructions for adjusting the tension of motor belts, tailstock alignment, and spindle alignment. The spindle alignment step involves several sub-steps, including loosening the bearing lock nut, tightening the bearing adjustment nut, testing the spindle by rotating it, and finally re-tightening the bearing

¹³<https://www.omg.org/spec/CMMN/1.1/>

¹⁴This machinery features a gear-driven mechanism, crucial for various metalworking tasks commonly used in metalworking industries such as turning and threading.

locking nut, all contributing to the overall spindle alignment step. The process of maintaining and adjusting gear head lathe machinery involves a series of steps (such as safety checks, adjustments, and alignments) that transition the machinery between different states (such as operational, under maintenance, and calibrated).

Assumptions. For the use case we assume a graph-structured representation and basic control flow patterns.

Core concepts. In terms of **control** this applies since a sequential process with steps that must be completed in a specific order is described which may include patterns such as sequence (WCP1). **Data** could include information about the state of the machinery, measurements taken during maintenance, and any data related to the adjustments made thus including patterns such as case data (WDP5), and environment data (WDP8). In terms of **resources**, we may represent information about the individuals performing the maintenance and possibly the tools or equipment used as resources. In terms of hierarchies between processes, in this use case, we need to represent the include relationship between processes. Each time someone performs maintenance on the gear head lathe machinery, a system could generate a **trace** of the process, documenting how each step in the maintenance process was carried out, including the spindle alignment sub-steps. In the **online** dimension, should the trace be generated automatically, the use case would require some infrastructure to monitor the unfolding of the process.

Tasks. Queries can serve multiple purposes, including the retrieval of specific processes based on criteria like process name (*i.e.*, process retrieval), the extraction of specific steps or sub-steps from processes (*i.e.*, step extraction), and facilitating the comparison and analysis of multiple processes (*i.e.*, cross-process analysis). Users can query for common steps, shared sub-steps, or similarities/differences between processes, enabling the identification of reusable components, standardisation of processes, or the recognition of best practices across different processes. Reasoning plays a crucial role, contributing to tasks like process validation by employing logical reasoning rules to detect inconsistencies, contradictions, or missing steps within a process. Reasoning also supports process execution by determining the appropriate order of steps and handling dependencies (*i.e.*, process execution).

3.2 Manufacturing data spaces

The *Manufacturing data spaces* use case involves the cataloguing of data in manufacturing settings, where the goal is to catalogue the data sets collected during manufacturing, in order for data scientists to access the appropriate data for downstream data analytics. In this use case, the position in the manufacturing process must be part of the metadata of the data sets to consider this context during searches. The manufacturing process follows a stamp-forming flow, involving steps like heating, pre-forming, stamping, cooling, de-moulding, and reworking, with potential variations and additional pre- and post-processing steps. The **data** dimension encompasses the representation of specific data sets collected during manufacturing. As goods are manufactured, data is generated, and specific traces and data are recorded.

Assumptions. We assume a graph-based hierarchical process representation. The hierarchy allows for introducing variations on the lower levels of the process while keeping the high-level flow the same. Data analytics may involve the time spent on different event instances traces, hence time stamps are required.

Core concepts. In terms of **control**, various patterns are essential, including linear sequences (WCP1), case data (WDP5), environment data (WDP8), parallel split (WCP2), synchronisation (WCP3), and hierarchy/subprocesses (H). The aspect of **data** is relevant, as input and output materials and their occurrences are modelled. The aspect of **resource** is of interest, if the person who conducted the experiments is modelled. The aspect of **trace** is paramount in this scenario, as past experiments are recorded in traces for further querying. In this manufacturing data spaces scenario, the exclusion of the **online** dimension is justified by focusing on analysing past experiments rather than performing the analyses in real-time.

Tasks. The main task is querying traces. Queries focus on various aspects of the data and metadata generated during process instances, such as those related to products meeting key performance indicators (KPIs) or those executed by specific individuals or on particular days. Reasoning is limited in the use case, however a need can be foreseen to find the difference between two hierarchical process models.

3.3 Healthcare and tele-rehabilitation

The *Healthcare and tele-rehabilitation* use case involves modelling patient states following physical rehabilitation treatments, such as post-operative recovery, home-based reeducation or post-stroke physiotherapy. Traditional rehabilitation requires intensive intervention by healthcare professionals and may not adequately adapt to individual patient characteristics, context, or motivation. Digital rehabilitation addresses these challenges by offering personalised therapies – such as rehabilitation exercises – using sensors to monitor exercise performance and physiological markers. The patient’s process is modelled as a set of states, with transitions triggered by their own decisions and influenced by a digital assistant’s suggestions. Data gathered from wearable and environmental devices guide recommendations, such as resting, adjusting exercise intensity, or modifying the overall treatment based on the patient’s characteristics, motivation, and emotional status.

Assumptions. The scenario assumes representation of events and processes as graphs, and temporal order among events related to the patients. For instance, exercises can be represented as events that follow a temporal order, *e.g.*, `StartExercising SEQ ExerciseEasy` would represent a sequence of two events: the patient starting to exercise, followed by an easy exercise.

Core concepts. The different steps in the use case can be represented as events (*e.g.*, `EasyExercise`), while states can also be used to represent activity stages of a patient (*e.g.*, `resting`). Thus, connections among states and events could be possible. For instance an expression as `StopExercise → resting` can represent temporal order among an event and a state. Moreover, occurrences are needed in order to characterise particular instances of events, *i.e.*, specific exercises performed by a specific patient.

Tasks. As part of the use case different tasks need to be performed. Among these, continuous queries have to be performed over the events captured by the motion and activity sensors. These queries may request sensor data related to exercise performance or the patient’s physical status within specific time boundaries. These requests are expressed using SPARQL-based queries, for instance monitoring the number of exercise movements performed as well as the time spanned in and between exercises. Analysis of exercise progression and order can also be performed, *e.g.*, identifying the sequences of exercises over time. Furthermore, queries can also verify the sequence of correct and incorrect exercises performed during a session. These queries require the notion

of *now*, *i.e.*, the moment used as a reference for computing time windows, and/or evaluating the occurrence of events over time. Query results can be continuously generated and used to create higher-level events through a stream reasoner, such as determining if a patient is fatigued based on movement quality, breathing and heartbeat. Additionally, the patient's treatment pathway can be modelled over time as a process, enabling the detection of non-compliance or continuous under-performance among patients, among other insights.

3.4 Domain-aware simulation

In the *Domain-aware simulation* use case, simulators must schedule and execute tasks in an initially unknown order to transform an initial state into a result state. To access this simulation result, they must also provide a means to query the result state and the sequence of executed tasks leading to it. A simulation task is modelling a state change according to some process in the domain of the simulation. Thus, they are inherently domain-specific and must have precisely described conditions for process initiation. These tasks can be modelled ontologically to allow the simulator to understand both their ontological description (conditions for process initiation) and computational description (changes needed to transform the current state).

While the domain modelling is often implicit in the simulation software, it can be made explicit. The simulator queries its state for possible steps at each simulation step, executes them in parallel for all parts, and analyses whether their changes trigger further tasks in the same simulation step. This can be done either by querying for newly enabled processes based on the state or by modelling dependencies between processes.

The geological simulator of Qu et al. [92], which simulates geological processes such as deposition of material and subsequent chemical process related to hydrocarbon generation, is an example where the domain modelling is made explicit: Such a domain-aware simulator relates its (internal) state to (external) geological descriptions, such as composition of geological layers, and its tasks to geological processes, such as deposition.

Assumptions. The use case assumes a graph representation of the domain, as well as a notion of sequence – while a transition is *realised* in the simulator, it must be *described and recorded* outside of it.

Core concepts. Concerning the representation of **control**, one must, beyond basic patterns like sequences (WCP1), express that all possible tasks must be checked for applicability in some order and may be executed concurrently (*e.g.*, Interleaved Parallel Routing WCP17, Interleaved-Routing WCP40) is necessary. For example, geological processes such as deposition can be concurrently active in different areas, such as erosion on two different mountains, or concurrently in the same area, such as heating of organic material and compaction, both due to increased pressure.

Representing **control** also affects data representation, as data may have cause-effect relationships between them, such as in Data-Based Routing (*e.g.*, Task Precondition Data - WDP35, Task Postcondition Data - WDP37, Data-based Task Trigger - WDP39), which should be able to interpret the simulated state as a knowledge graph.

The query tasks outlined below require an explicit representation of both the **trace**, especially for temporal properties of simulation runs and **data**, to examine the content of the simulation state at each point. In case of hyperproperties [24], *i.e.*, specification and comparison of multiple simulator runs, one also wants to represent, and query, multiple traces. It may, depending on the application, also be relevant to examine **resources**, *e.g.*, to examine the scheduling decisions of a concurrent simulator due to thread synchronisation or load balancing.

Tasks. The overall task involves executing the simulation and two additional reasoning and querying subprocesses after completion. First, it requires access to the final state and trace through queries to retrieve the simulation's results. Second, as intermediate steps, the simulator must execute membership queries to determine which tasks can run next. These queries may target either the current simulation state or the overall sequence of executed tasks, and they can express specifications over the simulation algorithm or domain-specific constraints on task sequences. In terms of the aforementioned geological simulator, such a query may retrieve all the parts of the state, where a certain geological process starts [92].

Beyond querying, one must be able to reason about the simulation runs and results with respect to the domain. This includes expressing temporal properties over single simulation runs to express temporal constraints. Potentially, one may need to investigate either *all* possible runs, *e.g.*, if certain parameters are not known, or compare multiple runs, to compare the effects of parameter changes. This is akin to model checking and expressing hyperproperties. Similarly, one may want to *query* multiple runs of the simulation.

3.5 Aircraft cockpit design

The *Aircraft cockpit design* use case focuses on ergonomic analyses of aircraft cockpits in virtual reality. Aircraft cockpits must be designed for the safe execution of pilot workflows. To support ergonomic analyses of aircraft cockpits during the early design phase when only CAD models are available, a workflow-aware analysis tool has been developed [62]. The tool enables the analysis and visualisation of pilot workflows in aircraft cockpits within a virtual reality environment. Hierarchical process models are employed to break down the processes pilots perform in aircraft cockpits. For example, take-off phases (*e.g.*, taxi, climb) are at a higher level, while physical pilot activities (*e.g.*, moving throttle, engaging autopilot) are at the leaf level. Data includes information related to aircraft cockpit design specifications, such as data structures in Virtual Reality derived from CAD models. During ergonomic analyses of aircraft cockpits, occurrences are generated to gain insights into the progress of pilot workflows and cockpit interactions.

Assumptions. We represent processes with hierarchies as graphs. The hierarchies allow us to cluster different activities according to domain practices, *e.g.*, to talk about certain phases of operation (taxi, climb) that involve different activities. Not all parts of the system are synchronised, but the process runtime aims to be fast to follow what happens in the different other subsystems. The process runtime does so by sending HTTP GET requests to the other subsystems in order to observe different states, which those subsystems are in. Thus, the state *now* can be determined by sending HTTP GET requests to all resources in all subsystems (or a subset thereof) and taking the union of the resource state representations in the responses.

Core concepts. In the process model, we tie together activities (*i.e.*, events). Conditions on system state allow to track the progress of occurrences and inform decisions which branch to follow in a conditional split. In terms of **control**, essential patterns include hierarchical modelling (H), sequential (WCP1), parallel (WCP2, WCP3), conditional (WCP4), any order (WCP40), and modelling conditions for splits and joins. Activity completion is determined using queries to state **data** (*e.g.*, SPARQL ASK queries) (WDP16, WDP35, WDP37, WDP40). Execution also requires modelling the tasks associated with activities (*e.g.*, unsafe HTTP requests) (WDP15). Similarly, conditions are evaluated on states. On top, execution requires support for **online** processing, *i.e.*, managing multiple process instances alongside models. **Resources** that drive the progress are not explicitly modelled, as there is only one agent (*i.e.*, the pilot) under consideration at a time. The aircraft simulation also in a way can drive the progress by corresponding state changes represented in the environment, but is also not explicitly modelled. The **trace** aspect has also only had a minor role, where timestamps have been attached to finished instances of activities.

Tasks. Querying is an integral part of the execution semantics, helping find tasks for activities to be executed or determining the next steps based on the control flow. RDFS Reasoning on the process model is required to ascertain when complex sub-processes have been completed or to identify the next task or all upcoming tasks within the hierarchical model's leaf level.

3.6 Summary of requirements for representation of processes

In addition to representing processes and related concepts to informally specify processes, as done for example in the various graphical notations, many scenarios require means to carry out tasks over the process descriptions. Thus, a formal representation of processes and related concepts is mandatory.

Table 3 gives an overview of the workflow patterns present in each scenario, providing a quick reference to understand how these patterns are distributed across the scenarios. Starting from the scenario *Manufacturing process documentation*, which covers only two workflow patterns, it is evident that the level of difficulty of each subsequent scenario increases. Therefore, as we progress to scenario *Domain-aware simulation*, a more complex use case, we can anticipate that it will have a broader coverage of workflow patterns. This incremental approach allows for a systematic exploration of workflow patterns, gradually building up complexity and understanding as we advance through the scenarios. Both control and data patterns are essential for designing comprehensive workflow models. They provide a structured way to represent the control logic and data interactions within a workflow.

3.7 Summary of requirements for querying and reasoning tasks

All scenarios assume that information about processes and related concepts are represented as graphs. The requirements of the previous scenarios regarding querying and reasoning can be grouped according to the five dimensions. Please note that the representation influences the kind of operations that can be applied on the representation (and vice versa). In the following, we thus outline and summarise typical operations that can be carried out over the state component, *i.e.*, mainly the data and resource dimensions in our classification, and typical operations that can be done on the temporal component, *i.e.*, mainly the control, trace and online dimensions.

If we only operate on the graph structure of the process descriptions, we can use standard graph-based data models (such as RDF) as basis for querying using SPARQL, possibly in conjunction with ontologies (RDFS or OWL). The requirements from the various scenarios regarding querying and reasoning are summarised and generalised in Table 4. In Table 4, we aim to simplify the presentation by concentrating on the requirements that directly align with the primary objectives addressed by the use case scenarios. Requirements can be further combined, particularly concerning the combination of process, data and online.

Next to performing reasoning on just the graph representation (*e.g.*, using RDF or RDFS semantics, using rules, using OWL DL (direct) semantics), some scenarios also require a way to perform reasoning on and with the temporal structure. Many graph query languages have means to deal with sequences, so basic retrieval and query tasks should be possible, including reachability. When applying reasoning, additional queries can be supported, for example for declaring a property transitive (*e.g.*, the next property) or declare two properties the inverse of each other (*e.g.*, next and previous).

More elaborate operations on temporal structure are better supported with dedicated languages of temporal logics. However, only the *Healthcare and tele-rehabilitation* and the *Domain-aware simulation* scenarios require query and reasoning tasks beyond the standard tasks on graph-structure representations. The scenario *Healthcare and tele-rehabilitation* is special concerning the

■ **Table 3** Dimensions of processes and scenarios that require them. We give specific features for the dimensions. If there exists a corresponding Workflow Pattern for the feature in [102], we give the name and its shorthand (*WxPn*). Features without a workflow pattern are given a single-letter shorthand.

	ID	Name	Scenario
Control	WCP1	Sequence: steps are executed in a linear order, one after another.	Manufacturing process documentation (Sec. 3.1) Manufacturing data spaces (Sec. 3.2) Healthcare and tele-rehabilitation (Sec. 3.3) Domain-aware simulation (Sec. 3.4) Aircraft cockpit design (Sec. 3.5)
	WCP2/3	Parallel Split: the divergence of a branch into two or more parallel branches each of which executes concurrently. Synchronisation: activities are executed in parallel, allowing multiple paths of execution to occur simultaneously.	Manufacturing data spaces (Sec. 3.2) Aircraft cockpit design (Sec. 3.5)
	WCP4	Exclusive Choice: the divergence of a branch into two or more branches such that when the incoming branch is enabled, the thread of control is immediately passed to precisely one of the outgoing branches based on a selection mechanism.	Aircraft cockpit design (Sec. 3.5)
	WCP17	Interleaved Parallel Routing: a set of tasks has a partial ordering that defines the requirements for their execution order.	Domain-aware simulation (Sec. 3.4)
	WCP23	Transient Trigger: ability for a task instance to be triggered by a signal from another part of the process or the environment.	Healthcare and tele-rehabilitation (Sec. 3.3)
	WCP40	Interleaved Routing: each member of a set of tasks must be executed once. They can be executed in any order but no two tasks can be executed at the same time.	Domain-aware simulation (Sec. 3.4), Aircraft cockpit design (Sec. 3.5)
	H	Hierarchical Modelling	Manufacturing process documentation (Sec. 3.1) Manufacturing data spaces (Sec. 3.2) Aircraft cockpit design (Sec. 3.5)
	Online	O	Online processing
Trace	T	Trace	Manufacturing process documentation (Sec. 3.1) Manufacturing data spaces (Sec. 3.2) Domain-aware simulation (Sec. 3.4)
Data	WDP1/3	Task data: data elements can be defined by tasks accessible only within the context of individual execution instances of that task. Scope data: data elements can be defined which are accessible by a subset of the tasks in a case.	Healthcare and tele-rehabilitation (Sec. 3.3)
	WDP5	Case Data: where data elements which are specific to a process instance or case are supported.	Manufacturing process documentation (Sec. 3.1) Manufacturing data spaces (Sec. 3.2)
	WDP8	Environment Data: allows data elements from the external operating environment to be accessed by components of processes during execution.	Manufacturing process documentation (Sec. 3.1) Healthcare and tele-rehabilitation (Sec. 3.3), Manufacturing data spaces (Sec. 3.2)
	WDP15/16	Task to Environment Pull: the ability of a task to request data elements from resources or services in the operational environment. Task to Environment Push: the ability of a task to initiate the passing of data elements to a resource or service in the operating environment.	Aircraft cockpit design (Sec. 3.5)
	WDP35/37	Task precondition Data Value: can be specified for tasks based on the value of specific parameters at the time of execution. Task postcondition data value: can be specified for tasks based on the value of specific parameters at the time of execution.	Domain-aware simulation (Sec. 3.4), Aircraft cockpit design (Sec. 3.5)
	WDP39	Data-based Task Trigger: provide the ability to trigger a specific task when an expression based on data elements in the process instance evaluates to true.	Domain-aware simulation (Sec. 3.4)
	WDP40	Data-based Routing: provides the ability to alter the control flow within a case based on the evaluation of data-based expressions.	Aircraft cockpit design (Sec. 3.5)
Resource	R	Resource	Manufacturing process documentation (Sec. 3.1) Manufacturing data spaces (Sec. 3.2)

■ **Table 4** List of requirements for querying and reasoning, grouped according to the five dimensions (Control, Trace, Data, Resource and Online), and pointing to the scenario where they appear.

Dimension	Requirement	Scenario
Control	Procedure Retrieval	All
	Steps Retrieval	All
	Sub-procedure Retrieval	Manufacturing documentation (Sec. 3.1) Manufacturing data spaces (Sec. 3.2) Aircraft cockpit design (Sec. 3.5)
Trace	Sequence Analysis	All
	Cross-procedure Analysis	All
Data	Retrieval of Domain Data	All
Resource	Retrieval of Personnel or Equipment	All
	Resource Allocation	Healthcare and tele-rehabilitation (Sec. 3.3) Domain-aware simulation (Sec. 3.4)
Online	Occurrence Retrieval	Healthcare and tele-rehabilitation (Sec. 3.3)
		Domain-aware simulation (Sec. 3.4)
		Aircraft cockpit design (Sec. 3.5)

representation of time. While the other scenarios only require a linear ordering, the querying and reasoning tasks in the *Healthcare and tele-rehabilitation* scenario require temporal queries with an operator defined over moving windows (*i.e.*, different time periods). The need for temporal logic specifications is the most clear in the scenario *Domain-aware simulation*. The query and reasoning tasks for one trace alone must be able to reason about safety and liveness properties, while comparing two traces is a natural setting for temporal logics.

4 Process description concepts

We start with a set of assumptions to scope our work on a possible core process ontology. The process ontology core should support a graph representation and temporal order and should operate in a web environment. For each assumption, we first give an overview of the concepts underlying a representation of processes and related concepts and then introduce a set of tasks that operate on the representations.

- **Graph Representation:** We assume a graph representation to describe the concepts of a possible core process ontology. In particular, for the description of processes and related concepts, we assume graphs according to the RDF recommendation of the W3C. Graphs provide flexibility in the descriptions of processes and related concepts, especially allowing for partial descriptions that just cover certain aspects and for incrementally adding descriptions. Depending on what is described in the graph, we can bring different methods to bear on the representations.
- **Time and Temporal Properties:** Next, we assume that we need to describe temporal properties of processes and related concepts. To support temporal reasoning, we need means to qualify the descriptions in terms of time. As a starting point, we assume discrete metric time, that is, we assign to each temporal entity a timestamp as integer. We could also assume a weaker notion of time, linear time, with just a partially ordered relation “happened-before” between temporal entities. Thus, with metric time, we can arrange temporal entities along a single global time line, leading to a sequence of temporal entities. With linear time, we do not necessarily have a single global time line, but may arrive at multiple time lines, leading to multiple sequences (connected via a “directly-follows” relation).

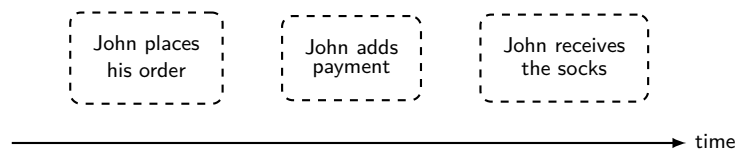
- The Web: The final assumption relates to infrastructure. The technologies around the web are widely deployed and can provide the infrastructure for exchanging representations and executing processes. Thus, we assume that the processes and related concepts we consider operate in a web environment. Combining graph representations with web architecture, we arrive at graph representations that are accessible via HTTP (Linked Data). In addition, on the web, multiple processes may run concurrently and communicate with each other. Thus, processes must be able to send and receive requests during execution. If we want to support runtime with some form of online processes, we need a dedicated 'now' point in the time line, representing the current point in time.

We now develop some of the core concepts for the representation of processes. Representing processes can be approached from different perspectives, considering the application domain. Given that the various fields have different central concerns and methods, the terminologies differ from field to field. Thus, in the following we attempt to identify concepts shared among different fields, used as a common foundation to represent processes.

We introduce these main concepts by means of an example, in the style of linguistics [43]. Let us consider a simple version of a purchasing process in an e-commerce setting.

John bought a pair of socks online last week. He first placed the order, then he paid and finally he received the socks from the retailer.

The example consists of three steps: John placing an order, John adding payment and John receiving the shipment. The different steps occur in a certain temporal order. Rather than using dates and wall clock time, or more abstractly integers to refer to different points in time, we can also build on a weaker notion of time. We have one relation “happened-before” [70] that we use also later in the temporal specifications (for example, in linear-time temporal logic). Linear (total) ordering of temporal entities is sufficient for most of the approaches we consider¹⁵. Figure 3 illustrates the temporal ordering of the various steps, *i.e.*, the sequence in which the steps are occurring.



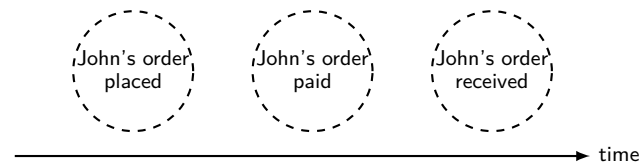
■ **Figure 1** The different steps of the purchasing process happen in sequential order.

We assume temporal entities, *i.e.*, entities which have some attachment to time. Temporal entities could be distinguished into describing “what is being the case” (states) or “what is happening” (non-states). For now we assume that anything that can be put into a temporal order is a temporal entity.

Next to the different steps, in a state-centric perspective the purchasing process – or rather the data associated to the purchase – is going through various **states**. First, John’s order is in the **Placed** state; next, John’s order is in the **Paid** state; and finally, John’s order is in the **Received** state.

The order object might also contain information about the product being purchased, in the example one pair of socks. Figure 2 illustrates the different states in a visual representation.

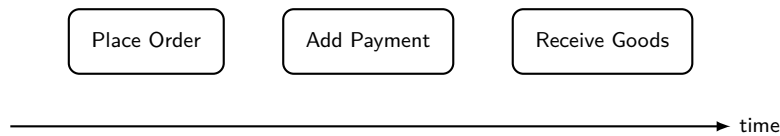
¹⁵ Although stream reasoning requires a more expressive way to represent time.



■ **Figure 2** The object associated with the purchasing process goes through various states as the process unfolds.

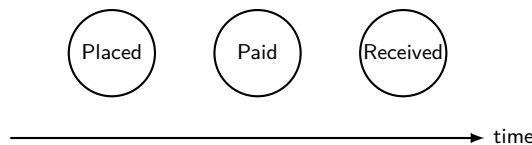
We now discuss the classification of temporal entities as particulars or universals [93]. When representing state, an established way to data modelling is the distinction between classes and instances. Together with various logical modelling constructs, we can refer to ontology languages such as RDF Schema or OWL.

When it comes to non-states (“what is happening”) we also want to make the distinction between particulars (the specific temporal entity where John ordered a pair of socks) and universals (the general temporal entity that describes all of the times when somebody is placing an order). Thus, we introduce the notion of **events**, which concern the level of universals, and **occurrences (of events)**, that concern the level of particulars. We denote as events the different steps (non-states, “what is happening”) on a terminology or schema level. Figure 3 shows the three steps on the level of universals. In the example, an occurrence of the **Place Order** event would be John placing an order for a pair of socks. An occurrence of the **Add Payment** event would be John paying via bank transfer; and for the **Receive Goods** event, an occurrence would be John receiving the socks via mail.



■ **Figure 3** The different steps of the purchasing process happen in sequential order.

Similarly, we can consider the different states on the level of universals. First, an order object is in the **Placed** state; next, the object is in the **Paid** state; and finally, the object is in the **Received** state.



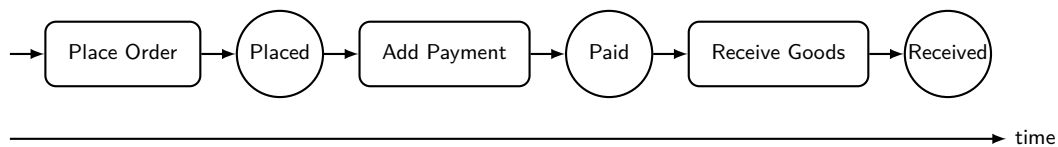
■ **Figure 4** The object associated with the purchasing process goes through various states as the process unfolds.

To sum up, in event-centric approaches, the constituent elements are occurrences of events, while in state-centric approaches, the constituent elements are instances of objects. A trace needs to include information to connect the elements, *e.g.*, via a case identifier.

Each **process** is an event (and each occurrence of a process is an occurrence of an event). The distinction between a process and an event comes from whether the behaviour of the event is further described and separated into different (or events). Until now, we have only considered

sequences within processes, but more elaborate control constructs are possible. For example, non-deterministic choice and parallelism (to cater for different agents that drive steps) are other control flow patterns that we can include into our process modelling approach.

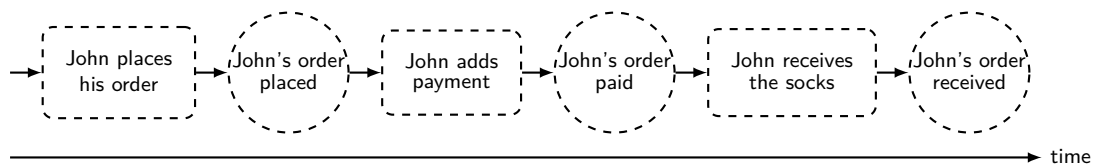
So far, we have only assumed a temporal order in the events (or states) that make up a process. We now introduce the notion of a control flow that ties together the events. In the simplest case of sequences, we can tie together the events via a “next” relation (immediately-follows), indicating (in our example) that the next event after Place Order is Add Payment, and the next event after Add Payment is Receive Goods. Figure 5 illustrates in a graphical notation in the style of Petri nets. Circles represent states (related to places in Petri nets) and rectangular shapes represent activities (related to transitions in Petri nets). Other ways to string together events constituting a process are possible and are discussed later.



■ **Figure 5** The temporal entities of the purchasing process in temporal order with connections indicating the control flow. Boxes denote events and circles denote states (in the style of Petri nets).

Business processes and workflows use the notion of an event log, *i.e.*, time-stamped occurrences of events together with an associated “case identifier”. Instead of a process “instance”, which is commonly used in the class/instance distinction, we introduce the concept of an **occurrence of a process**. To be able to capture the history of processes, we introduce the notion of **traces**. Traces record the specific occurrences of an event (and possibly of states) together with the temporal order in which they occurred.

Figure 6 shows a trace involving both the occurrences of events and the respective states.



■ **Figure 6** The temporal entities (occurrences of events and states) of the purchasing process occurrence represented as trace. Dashed boxes denote event occurrences and dashed circles denote state occurrences (in the style of Petri nets). Lines with arrows denote a temporal dependency.

Events might happen without an agent responsible for the event. In the business process community the non-state (“what is happening”) entities are known as activities on a terminology or schema level; but the notion of activity implies some sort of agent that carries the activity out. Although in the example we can identify the different people (or roles) that carry out a step, in our conceptualisation we aim to be as general as possible and do not assume agents that are causing a step to happen.

Nevertheless, many scenarios require the identification of the agents (or tools) involved in a process. Thus, we introduce the notion of **resources**, that is, the agents being responsible for carrying out an event (or being patient/participant in an event). We can identify resources both at the level of particulars or universals (*e.g.*, John vs. customer role).

Next to sequences of “directly-follows” temporal entities, we can also introduce the notion of choice. Extending our example, we could say that the next event after Place Order is Add Payment via Bank Transfer or Add Payment via Credit Card.

For now, we do not further specify the conditions attached to the choice and assume random (*i.e.*, non-deterministic) choice. In addition, for a more complete description of a process, we could assume multiple processes that interact with each other. In the purchasing process example, we would have John’s view on the process and the retailer’s view. After John pays, he awaits delivery of the socks. After the retailer receives John’s payment, the retailer ships John’s socks. Thus, we need a way to represent multiple (possibly concurrent) processes.

A final construct that is commonly found in process representations is that of hierarchical decomposition, which implies a hierarchical relationship between a main process and other processes. In our example, we could decompose the **Add Payment** event into a choice between **Add Payment via Bank Account** and **Add Payment via Credit Card**. Furthermore, there can be different relationships in hierarchical modelling. The relationship may indicate either i) a dependency between a main process and other processes, *i.e.*, when the main process is executed, simultaneous execution of other processes is required in the meantime, or ii) a generalisation between a main process and other processes, *i.e.*, when there are variations in terms of the activities and the number of activities between the main process and the other processes. Thus, hierarchical modelling may facilitate the organisation of complex processes.

5 Challenges for a core process vocabulary

Based on the scenarios outlined above, we can identify a number of challenges for representing processes and related concepts and perform certain reasoning tasks over the representation. We structure our discussion into challenges related to representing processes as graphs, representing queries and formulas and reasoning with process descriptions and formulas.

5.1 Representing processes as graphs

A core process vocabulary should be broadly applicable to describe processes and represent a consensus from different fields on the abstractions and concepts as well as the terminology. All our use cases take a discrete-time perspective, which we can take as a fundamental assumption going forward. We have already illustrated why we need support for both state-centric and activity-centric perspectives. Similarly, all of our scenarios benefit from a representation based on graphs. We have started to work out required concepts from first principles and distil commonalities between different fields based on the five dimensions, which we will discuss further in the remainder of the section. Still, to arrive at a fixed set of vocabulary terms requires a deeper alignment of concepts and more study of the representations of processes in related fields.

In a process core vocabulary, we have to decide which terms to include and which terms to possibly relegate to extensions. That is, a process core vocabulary should be layered, where additional tasks can be supported if the underlying representation is gradually extended. In the control dimension, the question is what patterns to support next to the sequence pattern (WCP1) expressing temporal order. More patterns might be useful for a core vocabulary. Especially the any order pattern (WCP40) could be rather directly supported, as well as the exclusive choice pattern (WCP4), given that both patterns relate to choice and do not require parallelism. Parallelism (WCP2 and WCP3) might be a worthwhile pattern to support, especially when the core vocabulary should include support for multiple roles (see the resources dimension). A process core could support hierarchical modelling (the “H” pattern in Table 3) to be able to modularise the process representations and thus help to keep an overview in large and complex processes and provide a mechanism for iterative modelling. How to provide useful primitives for specialisation or generalisation is an open question.

Representing occurrences and traces of processes is also a rather fundamental pattern, so fundamental in fact that the business process community just silently assumes such a primitive without explicitly identifying the pattern. We think that the particular/universal distinction in the context of processes is important, mirroring the type/instance distinction in ontology languages. Thus, a core vocabulary should make the particular/universal distinction for events and processes as well. What effects the particular/universal distinction on the side of events and processes should have on the modelling of data items (and vice versa) is another open question.

Regarding the data dimension, the representation with graphs as syntax allows for more flexibility compared to a dedicated language with a grammar of its own. In particular, a graph representation allows for freely extending the graphs with additional data, such as annotations on the level of the process descriptions, and enables the possibility of querying and deductive reasoning. Thus, a core vocabulary should include support for data, in particular environment data (WDP8), and integration with graph-structured data to be able to support data integration. For example, mappings to other vocabularies (*e.g.*, PROV-O) could be provided.

The representation of agents or roles involved in the process is required. Whether it is enough to have the flexibility of graphs to allow for such descriptions, or whether we need dedicated terms as part of a process vocabulary remains to be seen.

5.2 Representing queries and formulas

Representing processes as graphs using a shared vocabulary provides already some benefits, such as performing query evaluation and certain reasoning tasks on the graph representation. The scenarios identified various possible querying and reasoning tasks. Overall, the challenge is that the different querying and reasoning tasks should be possible to perform on a unified representation. In particular, a major challenge is to combine the purely graph-structured process representations with specifications of temporal properties for querying and reasoning. To support querying and reasoning tasks that go beyond checking satisfiability on the graph representation, we require means to encode queries and logical formulas pertaining to temporal properties.

If we assume a framework of time based on temporal ordering (“happened-before”), we could use temporal logics that operate over the linear order for specification of temporal properties to encode restrictions on temporal order. In particular, Linear-time Temporal Logic (LTL) or Computational Tree Logic (CTL) are temporal logics that can operate over linear time (LTL) or branching time (CTL). A challenge is that formulas written in temporal logics directly can be difficult to understand. Thus, a core process vocabulary could use higher-level abstractions for temporal specification patterns, such as precedence and cause and effect [39]. Using such high-level specifications presents an opportunity to increase the flexibility when modelling processes or temporal properties of traces. For example, instead of directly representing processes (*e.g.*, first A, then directly followed by B), users could use (weaker) temporal properties to give a higher-level representation of temporal structure (*e.g.*, A precedes B). More research is required to investigate the trade-offs regarding modelling flexibility and computational complexity. An additional challenge using the formalisms of temporal logics is how to bring together the state perspective and event perspective, as temporal logic formalisms often prefer to specify properties on sequences of states (and not events) [23].

Another question for a core process vocabulary is whether we should go beyond the simple “happened-before” relation and assume a richer framework for time. For example, Allen’s interval algebra [3] supports time points but also intervals and defines relations between time points and intervals. Similarly, stream reasoning assumes time stamps that can be used within window operators akin to intervals. An underlying temporal logic to capture such constructs could be Metric Temporal Logic (MTL). Again, the challenge is to decide on whether the increased expressive power is required and worth the computational cost.

In general, representation and reasoning will involve trade-offs depending on the underlying logics used: while some constructs could be written down using a process core vocabulary, performing operations (with a desired semantics) over such constructs might be computationally expensive or even infeasible. Ideally, we would like to find a representation that can in principle support a broad variety of tasks, with fragments or profiles for specific tasks, such as temporal querying, planning and synthesis, formal verification or process mining.

5.3 Reasoning with process descriptions and formulas

The range of operations for reasoning with process descriptions and formulas span from relatively basic queries on the graph structure to more intricate queries concerning the temporal structure of process representations. These advanced queries essentially encompass operations akin to model checking and formal verification.

Regarding operations on the graph structure, there are two overarching objectives: query answering, that is, acquiring solutions to queries, and reasoning, involving the inference of new formulas when given a set of formulas or a graph. RDF, RDFS, and various OWL profiles in conjunction with SPARQL support querying and lightweight reasoning on the graph representation of objects and their associated properties, in particular those in the data and resource dimension. These operations rely on standard semantics all underpinned by the formal notion of satisfiability. A possible avenue for further research is to identify whether and to which extent reasoning on the graph structure might yield simple inferences on the temporal structure, although the specifics of this interaction remain a topic of exploration.

Of particular interest is the provision of support for operations on the temporal structure. A fundamental operation relates to checking the satisfiability of temporal specifications, which can be performed on process descriptions or traces. In addition, other operations are conceivable, such as finding processes that satisfy temporal specifications, akin to planning or synthesis. Conversely, another operation would be process mining, *i.e.*, identifying a process that can generate a given set of occurrences.

As part of a specification of the semantics of a core process vocabulary, generic logic formulas could be introduced that operate on the temporal structure independently of the domain-specific elements. These formulas can capture intricate concepts like “happened-before” or the temporal ordering of events and occurrences. Furthermore, domain-specific formulas can encode restrictions specific to a given domain, such as the requirement that an order can only be shipped once the payment has been completed.

Another open challenge involves the interplay between the semantics of objects and data items and the semantics of the temporal structure. Related is the connection between particulars, representing instances and occurrences, and universals, embodying classes and events. Combining state and event perspectives, both on the levels of particulars and universals, presents a challenge for the development of a semantics, which requires the integration of state descriptions, represented as objects and their properties in graph form, with event descriptions that influence these states. Given an occurrence of a process in a specific state and a sequence of events, one should be able to derive the states of the related objects and data items. Conversely, it should be possible to work backward: starting with a sequence of states for the data items, one should be able to deduce the sequence of events that led the data items to those states. Moreover, the possible triggering of events needs to be evaluated over the state, with the introduction of a condition language to express “guards”.

A graph-structured representation is advantageous for its flexibility in describing process-related knowledge. However, different use cases may require different semantic approaches. For instance, a data integration scenario could benefit from an open-world semantics, allowing partial process descriptions from multiple sources to be seamlessly combined. In contrast, a process execution use

case may require a closed-world semantics, particularly because it necessitates making assumptions about the initiation and completion of processes and the comprehensiveness of each step within a process. Decisions surrounding whether to open or close the world depend on the specific tasks at hand.

A final open question relates to specialisation of processes involving hierarchy. In such cases, a closed-world assumption could become useful. A closed-world assumption would be equally applicable when working with prototypes [25], as opposed to the classic class-instance distinction based on sets. Prototypes could also support partial process descriptions that can later be amalgamated or refined. Issues may arise when one considers formalisms with open world and closed world side by side and interacting, both on the graph representation and with the temporal component.

A process core vocabulary which supports all conceivable tasks is likely to be too expressive and thus prohibitively expensive in terms of computational complexity. Thus, we probably want to end up with a semantics for the entire vocabulary and define profiles of the vocabulary of different expressive power suitable for the intended task. One challenge then relates to the identification and investigation of profiles, *i.e.*, subsets of the vocabulary, that support relevant reasoning tasks, while avoiding that parts of the vocabulary that have different meaning depending on the context.

6 Conclusion

We have introduced core concepts related to representing processes from different perspectives (state-centric and event-centric) that can be integrated into a combined perspective. Our aspiration is that a process core vocabulary should be more broadly applicable than to only business processes, especially given that such a vocabulary would use a graph representation that can be flexibly extended. We believe that the development of a process core vocabulary (the syntax for processes and related concepts as well as temporal properties) in a graph structure and the development of an associated formal semantics have to go hand in hand and influence each other.

We plan to start with a (minimal) core vocabulary for representing processes, which later can be extended to cater for more elaborate use cases. While currently only a subset of the scenarios we have described require querying and reasoning on the temporal structure, the core vocabulary should support the specification of temporal properties, to later enable querying and reasoning using such temporal property specifications. A formal semantics for the temporal structure would enable temporal querying and reasoning functionality on the graph-structured representations.

To connect with the established temporal logics formalisation, we require a combination of temporal specification patterns with the graph representation of processes, for example via extraction of temporal logic models from graphs. Such an approach would provide a separation of concerns and would reuse temporal logics languages and reasoning through a clear interface to the knowledge representation languages and reasoning systems. However, many questions regarding the specifics of such an integration remain open.

While the finite-state machine and traces formalisation as well as various temporal logics are well established, for the (non-technical) terms related to the level of knowledge representation (*e.g.*, for the resource dimension) there is currently no consensus across communities. Our paper is to be seen as a first suggestion of which concepts would go into a broadly applicable vocabulary for processes and related concepts as well as temporal properties. We plan to continue experimentation with process descriptions in a variety of use cases, both for representing processes and reasoning with process descriptions in first implementations based on existing systems (*i.e.*, model checkers, automated planners or process mining algorithms) to validate the representation in the vocabulary. We also encourage the community to experiment with process descriptions and report on their experiences.

References

- 1 Sudhir Agarwal, Sebastian Rudolph, and Andreas Abecker. Semantic description of distributed business processes. In *Proceedings of AI Meets Business Rules and Process Management*, pages 1–11. AAAI, 2008. URL: <http://www.aaai.org/Library/Symposia/Spring/2008/ss08-01-001.php>.
- 2 Newres Al Haider, Benoit Gaudin, and John Murphy. Execution trace exploration and analysis using ontologie. In *Proceedings of the Second International Conference on Runtime Verification (RV)*, volume 7186 of *LNCS*, pages 412–426. Springer, 2011. doi:10.1007/978-3-642-29860-8_33.
- 3 James F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843, 1983. doi:10.1145/182.358434.
- 4 João Paulo A. Almeida, Ricardo de Almeida Falbo, and Giancarlo Guizzardi. Events as entities in ontology-driven conceptual modeling. In *Proceedings of the 38th International Conference on Conceptual Modeling (ER)*, volume 11788 of *LNCS*, pages 469–483. Springer, 2019. doi:10.1007/978-3-030-33223-5_39.
- 5 Ilkay Altintas, Chad Berkley, Efrat Jaeger, Matthew Jones, Bertram Ludäscher, and Steve Mock. Kepler: an extensible system for design and execution of scientific workflows. In *Proceedings of the 16th International Conference on Scientific and Statistical Database Management*, pages 423–424, 2004. doi:10.1109/SSDM.2004.1311241.
- 6 Darko Anicic, Paul Fodor, Sebastian Rudolph, and Nenad Stojanovic. EP-SPARQL: a unified language for event processing and stream reasoning. In *Proceedings of the 20th International Conference on World Wide Web (WWW)*, pages 635–644. ACM, 2011. doi:10.1145/1963405.1963495.
- 7 Darko Anicic, Sebastian Rudolph, Paul Fodor, and Nenad Stojanovic. Stream reasoning and complex event processing in ETALIS. *Semantic Web Journal*, 3(4):397–407, 2012. doi:10.3233/sw-2011-0053.
- 8 Anupriya Ankolekar, Mark H. Burstein, Jerry R. Hobbs, Ora Lassila, David L. Martin, Sheila A. McIlraith, Srinu Narayanan, Massimo Paolucci, Terry R. Payne, Katia P. Sycara, and Honglei Zeng. DAML-S: semantic markup for web services. In *Proceedings of the First Semantic Web Working Symposium (SWWS)*, pages 411–430, 2001.
- 9 Amina Annane, Mouna Kamel, and Nathalie Aussenac-Gilles. Comparing business process ontologies for task monitoring. In *Proceedings of the 12th International Conference on Agents and Artificial Intelligence (ICAART)*, pages 634–643. SCITEPRESS, 2020. doi:10.5220/0008978706340643.
- 10 Mattia Atzeni and Maurizio Atzori. CodeOntology: RDF-ization of source code. In *Proceedings of the 16th International Semantic Web Conference (ISWC)*, volume 10588 of *LNCS*, pages 20–28. Springer, 2017. doi:10.1007/978-3-319-68204-4_2.
- 11 Jie Bao, Elisa Kendall, Deborah McGuinness, and Peter Patel-Schneider. OWL 2 Web Ontology Language Quick Reference Guide. Recommendation, W3C, 2009. Latest version available at <https://www.w3.org/TR/owl2-quick-reference/>. URL: <https://www.w3.org/TR/2009/REC-owl2-quick-reference-20091027/>.
- 12 Davide Barbieri, Daniele Braga, Stefano Ceri, Emanuele Della Valle, Yi Huang, Volker Tresp, Achim Rettinger, and Hendrik Wermser. Deductive and inductive stream reasoning for semantic social media analytics. *IEEE Intelligent Systems*, 25(6):32–41, 2010. doi:10.1109/MIS.2010.142.
- 13 Davide Francesco Barbieri, Daniele Braga, Stefano Ceri, Emanuele Della Valle, and Michael Grossniklaus. C-SPARQL. In *Proceedings of the 18th International Conference on World Wide Web (WWW)*. ACM Press, 2009. doi:10.1145/1526709.1526856.
- 14 Piergiorgio Bertoli, Francesco Corcoglioniti, Chiara Di Francescomarino, Mauro Dragoni, Chiara Ghidini, and Marco Pistore. Semantic modeling and analysis of complex data-aware processes and their executions. *Expert Systems with Applications*, 198:116702, 2022. doi:10.1016/j.eswa.2022.116702.
- 15 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. doi:10.3233/A0-210259.
- 16 Daniel Brand and Pitro Zafropulo. On communicating finite-state machines. *Journal of the ACM*, 30(2):323–342, 1983. doi:10.1145/322374.322380.
- 17 Dan Brickley and Ramanathan Guha. RDF Schema 1.1. Recommendation, W3C, 2014. Latest version available at <https://www.w3.org/TR/rdf-schema/>. URL: <https://www.w3.org/TR/2014/REC-rdf-schema-20140225/>.
- 18 Anila Sahar Butt and Peter Fitch. ProvONE+: A provenance model for scientific workflows. In *Proceedings of the 21st International Conference on Web Information Systems Engineering (WISE)*, volume 12343 of *LNCS*, pages 431–444. Springer, 2020. doi:10.1007/978-3-030-62008-0_30.
- 19 Liliana Cabral, Barry Norton, and John Domingue. The business process modelling ontology. In *Proceedings of the 4th International Workshop on Semantic Business Process Management (SBPM)*, pages 9–16. ACM, 2009. doi:10.1145/1944968.1944971.
- 20 Jean-Paul Calbimonte, Oscar Corcho, and Alasdair J. G. Gray. Enabling ontology-based access to streaming data sources. In *Proceedings of the 9th International Semantic Web Conference (ISWC)*, volume 6496 of *LNCS*, pages 96–111. Springer, 2010. doi:10.1007/978-3-642-17746-0_7.
- 21 Jean-Paul Calbimonte, Jose Mora, and Oscar Corcho. Query rewriting in RDF stream processing. In *Proceedings of the 13th European Semantic Web Conference (ESWC)*, volume 9678 of *LNCS*, pages 486–502. Springer, 2016. doi:10.1007/978-3-319-34129-3_30.
- 22 Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Riccardo Rosati. Actions and pro-

- grams over description logic knowledge bases: A functional approach. In *Knowing, Reasoning, and Acting: Essays in Honour of Hector J. Levesque*. College Press, 2011.
- 23 Marsha Chechik and Dimitrie O. Păun. Events in property patterns. In *Proceedings of 5th and 6th International SPIN Workshops*, volume 1680 of *LNCS*, pages 154–167. Springer, 1999. doi:10.1007/3-540-48234-2_13.
 - 24 Michael R. Clarkson, Bernd Finkbeiner, Masoud Koleini, Kristopher K. Micinski, Markus N. Rabe, and César Sánchez. Temporal logics for hyperproperties. In *Proceedings of the Third International Conference on Principles of Security and Trust POST*, volume 8414 of *LNCS*, pages 265–284. Springer, 2014. doi:10.1007/978-3-642-54792-8_15.
 - 25 Michael Cochez, Stefan Decker, and Eric Prud'Hommeaux. Knowledge representation on the web revisited: the case for prototypes. In *Proceedings of the 15th International Semantic Web Conference (ISWC)*, volume 9981 of *LNCS*, pages 151–166. Springer, 2016. doi:10.1007/978-3-319-46523-4_10.
 - 26 Simon Cox and Chris Little. Time Ontology in OWL. Recommendation, W3C, 2017. Latest version available at <https://www.w3.org/TR/owl-time/>. URL: <https://www.w3.org/TR/2017/REC-owl-time-20171019/>.
 - 27 Richard Cyganiak, David Wood, and Markus Lanthaler. RDF 1.1 Concepts and Abstract Syntax. Recommendation, W3C, 2014. Latest version available at <https://www.w3.org/TR/rdf11-concepts/>. URL: <https://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/>.
 - 28 Donald Davidson. The logical form of action sentences. In *The Logic of Decision and Action*, pages 81–95. University of Pittsburgh Press, 1967. doi:10.1093/oso/9780195136975.003.0036.
 - 29 Donald Davidson. Events and particulars. *Noûs*, pages 25–32, 1970. doi:10.2307/2214289.
 - 30 Camila Zacché de Aguiar, Ricardo de Almeida Falbo, and Vítor E. Silva Souza. OOC-O: A reference ontology on object-oriented code. In *Proceedings of the 38th Conference on Conceptual Modeling*, volume 11788 of *LNCS*, pages 13–27. Springer, 2019. doi:10.1007/978-3-030-33223-5_3.
 - 31 Wellington Moreira de Oliveira, Daniel de Oliveira, and Vanessa Braganholo. Provenance analytics for workflow-based computational experiments: A survey. *ACM Computing Surveys*, 51(3):53:1–53:25, 2018. doi:10.1145/3184900.
 - 32 Daniele Dell'Aglio, Minh Dao-Tran, Jean-Paul Calbimonte, Danh Le Phuoc, and Emanuele Della Valle. A query model to capture event pattern matching in RDF stream processing query languages. In *Proceedings of the 20th European Knowledge Acquisition Workshop (EKAW)*, volume 10024 of *LNCS*, pages 145–162. Springer, 2016. doi:10.1007/978-3-319-49004-5_10.
 - 33 Daniele Dell'Aglio, Emanuele Della Valle, Jean-Paul Calbimonte, and Oscar Corcho. RSP-QL semantics. *International Journal on Semantic Web and Information Systems*, 10(4):17–44, 2014. doi:10.4018/ijswis.2014100102.
 - 34 Daniele Dell'Aglio, Emanuele Della Valle, Frank van Harmelen, and Abraham Bernstein. Stream reasoning: A survey and outlook. *Data Science*, 1:59–83, 2017. doi:10.3233/DS-170006.
 - 35 Andreas Diepenbrock, Florian Rademacher, and Sabine Sachweh. An ontology-based approach for domain-driven design of microservice architectures. In *INFORMATIK*, volume P-275 of *LNI*, pages 1777–1791. Gesellschaft für Informatik, 2017. doi:10.18420/in2017_177.
 - 36 Remco M Dijkman, Marlon Dumas, and Chun Ouyang. Semantics and analysis of business process models in BPMN. *Information and Software technology*, 50(12):1281–1294, 2008. doi:10.1016/j.infsof.2008.02.006.
 - 37 Crystal Chang Din, Leif Harald Karlsen, Irina Pene, Oliver Stahl, Ingrid Chieh Yu, and Thomas Østerlie. Geological multi-scenario reasoning. In *Proceedings of the 12th Norwegian Information Security Conference*. NIK: Norsk Informatikkonferanse, 2019. URL: <http://hdl.handle.net/11250/2633598>.
 - 38 Clemens Dubslaff, Patrick Koopmann, and Anni-Yasmin Turhan. Ontology-mediated probabilistic model checking. In *Proceedings of the 15th International Conference on Integrated Formal Methods (IFM)*, volume 11918 of *LNCS*, 2019. doi:10.1007/978-3-030-34968-4_11.
 - 39 Matthew B. Dwyer, George S. Avrunin, and James C. Corbett. Patterns in property specifications for finite-state verification. In *Proceedings of the 21st International Conference on Software Engineering (ICSE)*, pages 411–420, 1999. doi:10.1145/302405.302672.
 - 40 E Allen Emerson and Edmund M Clarke. Using branching time temporal logic to synthesize synchronization skeletons. *Science of Computer Programming*, 2(3):241–266, 1982. doi:10.1016/0167-6423(83)90017-5.
 - 41 Ronald Fagin, Joseph Y. Halpern, Yoram Moses, and Moshe Y. Vardi. Knowledge-based programs. *Distributed Computing*, 10(4):199–225, 1997. doi:10.1007/s004460050038.
 - 42 Alessio Ferrari and Maurice H. ter Beek. Formal methods in railways: A systematic mapping study. *ACM Computing Surveys*, 55(4):69:1–69:37, 2023. doi:10.1145/3520480.
 - 43 Antony Galton. Processes as patterns of occurrence. In *Process, Action, and Experience*, pages 41–57. Oxford University Press, 2018. doi:10.1093/oso/9780198777991.003.0003.
 - 44 Aldo Gangemi, Silvio Peroni, David M. Shotton, and Fabio Vitali. The publishing workflow ontology (PWO). *Semantic Web*, 8(5):703–718, 2017. doi:10.3233/sw-160230.
 - 45 Daniel Garijo and Yolanda Gil. A new approach for publishing workflows: abstractions, standards, and linked data. In *Proceedings of the 6th Workshop on Workflows in Support of Large-scale Science (WORKS)*, pages 47–56, 2011. doi:10.1145/2110497.2110504.
 - 46 Daniel Garijo and Yolanda Gil. Augmenting PROV with plans in P-Plan: scientific processes as linked data. In *Proceedings of the Second International Workshop on Linked Science 2012 -*

- Tackling Big Data*, volume 951 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2012. URL: <https://ceur-ws.org/Vol-951/paper6.pdf>.
- 47 Daniel Garijo, Yolanda Gil, and Óscar Corcho. Abstract, link, publish, exploit: An end to end framework for workflow sharing. *Future Generation Computer Systems*, 75:271–283, 2017. doi: 10.1016/j.future.2017.01.008.
 - 48 Dragan Gašević and Vladan Devedžić. Petri Net ontology. *Knowledge-Based Systems*, 19(4):220–234, 2006. doi:10.1016/j.knosys.2005.12.003.
 - 49 Giuseppe De Giacomo, Yves Lespérance, and Hector J. Levesque. ConGolog, a concurrent programming language based on the situation calculus. *Artificial Intelligence*, 121(1-2):109–169, 2000. doi:10.1016/s0004-3702(00)00031-x.
 - 50 Yolanda Gil, Varun Ratnakar, Ewa Deelman, Gaurang Mehta, and Jihie Kim. Wings for Pegasus: Creating large-scale scientific applications using semantic representations of computational workflows. In *Proceedings of the 22nd AAAI Conference on Artificial Intelligence*, pages 1767–1774. AAAI Press, 2007. doi:10.5555/1620113.1620127.
 - 51 Nathan Gruber and Birte Glimm. A comparative study of stream reasoning engines. In *Proceedings of the 20th European Semantic Web Conference (ESWC)*, volume 13870 of *LNCS*, pages 21–37. Springer, 2023. doi:10.1007/978-3-031-33455-9_2.
 - 52 Ramanathan V. Guha, Dan Brickley, and Steve Macbeth. Schema.org: evolution of structured data on the web. *Communications of the ACM*, 59(2):44–51, 2016. doi:10.1145/2844544.
 - 53 Yuri Gurevich. *Evolving algebras 1993: Lipari guide*, pages 9–36. Oxford University Press, 1995.
 - 54 Reiner Hähnle and Marieke Huisman. Deductive software verification: From pen-and-paper proofs to industrial tools. In *Computing and Software Science*, volume 10000 of *LNCS*, pages 345–373. Springer, 2019. doi:10.1007/978-3-319-91908-9_18.
 - 55 Armin Haller, Emilia Cimpian, Adrian Mocan, Eyal Oren, and Christoph Bussler. WSMX - A semantic service-oriented architecture. In *Proceedings of the 2005 IEEE International Conference on Web Services (ICWS)*, pages 321–328. IEEE Computer Society, 2005. doi:10.1109/ICWS.2005.139.
 - 56 Charles Antony Richard Hoare. Communicating sequential processes. *Communications of the ACM*, 21(8):666–677, 1978. doi:10.1145/359576.359585.
 - 57 Richard Hull, Elio Damaggio, Fabiana Fournier, Manmohan Gupta, Fenno F. Terry Heath III, Stacy Hobson, Mark H. Linehan, Sridhar Maradugu, Anil Nigam, Piyawadee Sukaviriya, and Roman Vaculín. Introducing the guard-stage-milestone approach for specifying business entity lifecycles. In *Proceedings of the 7th International Workshop on Web Services and Formal Methods (WS-FM)*, volume 6551 of *LNCS*, pages 1–24. Springer, 2010. doi:10.1007/978-3-642-19589-1_1.
 - 58 Stefan Jablonski. Mobile: A modular workflow model and architecture. In *Working Conference on Dynamic Modelling and Information Systems*, 1994. URL: https://www.researchgate.net/publication/2720558_MOBILE_A_modular_workflow_model_and_architecture.
 - 59 Ni Jing. A PROV-O based approach to web content provenance. In *Proceedings of the 2015 International Conference on Logistics, Informatics and Service Sciences (LISS)*, pages 1–6. IEEE, 2015. doi:10.1109/LISS.2015.7369688.
 - 60 Tobias Käfer and Andreas Harth. Rule-based programming of user agents for linked data. In *Workshop on Linked Data on the Web (LDOW)*, volume 2073 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2018. URL: <https://ceur-ws.org/Vol-2073/article-05.pdf>.
 - 61 Tobias Käfer and Andreas Harth. Specifying, monitoring, and executing workflows in linked data environments. In *Proceedings of the 17th International Semantic Web Conference (ISWC)*, volume 11136 of *LNCS*, pages 424–440. Springer, 2018. doi:10.1007/978-3-030-00671-6_25.
 - 62 Tobias Käfer, Andreas Harth, and Sebastien Mamessier. Towards declarative programming and querying in a distributed cyber-physical system: The i-VISION case. In *Proceedings of the Second International Workshop on Modelling, Analysis, and Control of Complex CPS (CPSData)*, pages 1–6. IEEE, 2016. doi:10.1109/CPSData.2016.7496418.
 - 63 Tobias Käfer, Benjamin Jochum, Nico Abfalg, and Leonard Nürnberg. Specifying and executing user agents in an environment of reasoning and RESTful systems using the guard-stage-milestone approach. *Journal on Data Semantics*, 10(1-2):57–75, 2021. doi:10.1007/s13740-021-00123-0.
 - 64 Eduard Kamburjan and Crystal Chang Din. Runtime enforcement using knowledge bases. In *Proceedings of the International Conference on Fundamental Approaches to Software Engineering FASE*, volume 13991 of *LNCS*, pages 220–240. Springer, 2023. doi:10.1007/978-3-031-30826-0_12.
 - 65 Eduard Kamburjan, Vidar Norstein Klungre, and Martin Giese. Never mind the semantic gap: Modular, lazy and safe loading of RDF data. In *Proceedings of the 19th European Semantic Web Conference ESWC*, volume 13261 of *LNCS*, pages 200–216. Springer, 2022. doi:10.1007/978-3-031-06981-9_12.
 - 66 Eduard Kamburjan, Vidar Norstein Klungre, Rudolf Schlatte, Einar Broch Johnsen, and Martin Giese. Programming and debugging with semantically lifted states. In *Proceedings of the 18th European Semantic Web Conference ESWC*, volume 12731 of *LNCS*, pages 126–142. Springer, 2021. doi:10.1007/978-3-030-77385-4_8.
 - 67 Hirofumi Katsuno and Alberto O. Mendelzon. On the difference between updating a knowledge base and revising it. In *Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning (KR)*, pages 387–394. Morgan Kaufmann, 1991. doi:10.1017/cbo9780511526664.007.
 - 68 Gerhard Keller, Markus Nüttgens, and August-Wilhelm Scheer. *Semantische Prozessmodellierung auf der Grundlage "Ereignisgesteuerter Prozessketten (EPK)".* Veröffentlichungen des In-

- stituts für Wirtschaftsinformatik. - Saarbrücken : IWI, ISSN 1438-5678. - Vol. 89, 1992.
- 69 Agnes Koschmider and Andreas Oberweis. Ontology based business process description. In *Proceedings of the Open Interop Workshop on Enterprise Modelling and Ontologies for Interoperability*, volume 160 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2005. URL: <https://ceur-ws.org/Vol-160/paper12.pdf>.
 - 70 Leslie Lamport. Time, clocks, and the ordering of events in a distributed system. *Communications of the ACM*, 1978. doi:10.1145/359545.359563.
 - 71 Jean-Baptiste Lamy. Owlready: Ontology-oriented programming in Python with automatic classification and high level constructs for biomedical ontologies. *Artificial Intelligence in Medicine*, 80:11–28, 2017. doi:10.1016/j.artmed.2017.07.002.
 - 72 Ming-Che Lee, Ding Yen Ye, and Tzone I. Wang. Java learning object ontology. In *Proceedings of the 5th International Conference on Advanced Learning Technologies ICALT*, pages 538–542. IEEE, 2005. doi:10.1109/icalt.2005.185.
 - 73 Martin Leinberger. *Type-safe Programming for the Semantic Web*. PhD thesis, University of Koblenz and Landau, 2021. doi:10.3233/ssw52.
 - 74 Hector J. Levesque, Raymond Reiter, Yves Lespérance, Fangzhen Lin, and Richard B. Scherl. GOLOG: A logic programming language for dynamic domains. *The Journal of Logical Programming*, 31(1-3):59–83, 1997. doi:10.1016/S0743-1066(96)00121-5.
 - 75 David Martin, Massimo Paolucci, Sheila McIlraith, Mark Burstein, Drew McDermott, Deborah McGuinness, Bijan Parsia, Terry Payne, Marta Sabou, Monika Solanki, Naveen Srinivasan, and Katja Sycara. Bringing semantics to web services: The OWL-S approach. In *Proceedings of the First International Workshop on Semantic Web Services and Web Process Composition (SWSWPC)*, volume 3387 of *LNCIS*, pages 26–42. Springer, 2005. doi:10.1007/978-3-540-30581-1_4.
 - 76 Drew McDermott. A temporal logic for reasoning about processes and plans. *Cognitive Science*, 6(2):101–155, 1982. doi:10.1207/s15516709cog0602_1.
 - 77 Robin Milner. *A calculus of communicating systems*. Springer, 1980. doi:10.1007/3-540-10235-3.
 - 78 Robin Milner. *Communicating and mobile systems: the pi calculus*. Cambridge University Press, 1999.
 - 79 Michael zur Muehlen, Jeffrey V. Nickerson, and Keith D. Swenson. Developing web services choreography standards - the case of REST vs. SOAP. *Decision Support Systems*, 40(1):9–29, 2005. doi:10.1016/j.dss.2004.04.008.
 - 80 Yavor Nenov, Robert Piro, Boris Motik, Ian Horrocks, Zhe Wu, and Jay Banerjee. RD-Fox: A highly-scalable RDF store. In *Proceedings of the 14th International Semantic Web Conference (ISWC)*, volume 9367 of *LNCIS*, pages 3–20. Springer, 2015. doi:10.1007/978-3-319-25010-6_1.
 - 81 Daniel Oberle. How ontologies benefit enterprise applications. *Semantic Web*, 5(6):473–491, 2014. doi:10.3233/sw-130114.
 - 82 Daniel Oberle, Stephan Grimm, and Steffen Staab. An ontology for software. In *Handbook on Ontologies*, pages 383–402. Springer, 2009. doi:10.1007/978-3-540-92673-3_17.
 - 83 Daniel Oberle, Steffen Lamparter, Stephan Grimm, Denny Vrandečić, Steffen Staab, and Aldo Gangemi. Towards ontologies for formalizing modularization and communication in large software systems. *Applied Ontology*, 1(2):163–202, 2006. URL: <http://content.iospress.com/articles/applied-ontology/ao016>.
 - 84 J. Neil Otte, John Beverley, and Alan Ruttenberg. BFO: basic formal ontology. *Applied Ontology*, 17(1):17–43, 2022. doi:10.3233/A0-220262.
 - 85 Alexander Paar. *Zhi# - programming language inherent support for ontologies*. PhD thesis, Karlsruhe Institute of Technology, 2009. doi:10.5445/IR/1000019039.
 - 86 Dileep Kumar Pattipati, Rupesh Nasre, and Sreenivasa Kumar Puligundla. BOLD: an ontology-based log debugger for C programs. *Automated Software Engineering*, 29(1):2, 2022. doi:10.1007/s10515-021-00308-8.
 - 87 Maja Pesic, Helen Schonenberg, and Wil MP Van der Aalst. DECLARE: Full support for loosely-structured processes. In *Proceedings of the 11th IEEE International Enterprise Distributed Object Computing Conference (EDOC)*, pages 287–300. IEEE, 2007. doi:10.1109/EDOC.2007.14.
 - 88 Carl Adam Petri. *Kommunikation mit Automaten*. PhD thesis, University of Bonn, 1962.
 - 89 Danh Le Phuoc, Minh Dao-Tran, Josiane Xavier Parreira, and Manfred Hauswirth. A native and adaptive approach for unified processing of linked streams and linked data. In *Proceedings of the 10th International Semantic Web Conference (ISWC)*, volume 7031 of *LNCIS*, pages 370–388. Springer, 2011. doi:10.1007/978-3-642-25073-6_24.
 - 90 Amir Pnueli. The temporal logic of programs. In *Proceedings of the 18th Annual Symposium on Foundations of Computer Science (SFCS)*, pages 46–57. IEEE, 1977. doi:10.1109/SFCS.1977.32.
 - 91 Arthur N Prior. *Time and modality*. Oxford University Press, 1957. doi:10.2307/2216989.
 - 92 Yuanwei Qu, Eduard Kamburjan, Anita Torabi, and Martin Giese. Semantically triggered qualitative simulation of a geological process. *Applied Computing and Geosciences*, 21:100152, 2024. doi:10.1016/j.acags.2023.100152.
 - 93 Fabrício Henrique Rodrigues and Mara Abel. What to consider about events: A survey on the ontology of occurrents. *Applied Ontology*, 14(4):343–378, 2019. doi:10.3233/ao-190217.
 - 94 Dumitru Roman, Uwe Keller, Holger Lausen, Jos De Bruijn, Rubén Lara, Michael Stollberg, Axel Polleres, Cristina Feier, Christoph Bussler, and Dieter Fensel. Web service modeling ontology. *Applied Ontology*, 1(1):77–106, 2005. doi:10.1007/978-3-642-19193-0_7.
 - 95 Marco Rospocher, Chiara Ghidini, and Luciano Serafini. An ontology for the business process modelling notation. In *Proceedings of the 8th International Conference on Formal Ontology in Information Systems (FOIS)*, volume 267 of *Frontiers in Artificial Intelligence and Applications*,

- pages 133–146. IOS Press, 2014. doi:10.3233/978-1-61499-438-1-133.
- 96 Nick Russell, Wil M. P. van der Aalst, and Arthur H. M. ter Hofstede. *Workflow Patterns: The Definitive Guide*. MIT Press, 2016. doi:10.7551/mitpress/8085.001.0001.
- 97 Barry Smith. Classifying processes: an essay in applied ontology. *Classifying Reality*, pages 101–126, 2013.
- 98 Graeme Stevenson and Simon Dobson. Sapphire: Generating Java runtime artefacts from OWL ontologies. In *Proceedings of the CAiSE Workshops*, volume 83 of *Lecture Notes in Business Information Processing*, pages 425–436. Springer, 2011. doi:10.1007/978-3-642-22056-2_46.
- 99 Oliver Thomas and Michael Fellmann. Semantic EPC: enhancing process modeling using ontology languages. In *Proceedings of the Workshop on Semantic Business Process and Product Lifecycle Management (SBPM)*, volume 251 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2007. URL: <https://ceur-ws.org/Vol-251/paper9.pdf>.
- 100 Riccardo Tommasini, Yehia Abo Sedira, Daniele Dell’Aglia, Marco Balduini, Muhammad Intizar Ali, Danh Le Phuoc, Emanuele Della Valle, and Jean-Paul Calbimonte. VoCaLS: Vocabulary and catalog of linked streams. In *Proceedings of the 17th International Semantic Web Conference (ISWC)*, volume 11137 of *LNCS*, pages 256–272. Springer, 2018. doi:10.1007/978-3-030-00668-6_16.
- 101 Daniele Turi, Paolo Missier, Carole A. Goble, David De Roure, and Tom Oinn. Taverna workflows: Syntax and semantics. In *Proceedings of the Third International Conference on e-Science and Grid Computing*, pages 441–448. IEEE, 2007. doi:10.1109/E-SCIENCE.2007.71.
- 102 Wil M. P. van der Aalst, Arthur H. M. ter Hofstede, Bartek Kiepuszewski, and Alistair P. Barros. Workflow patterns. *Distributed and Parallel Databases*, 14(1):5–51, 2003. doi:10.1023/A:1022883727209.
- 103 Wil M. P. van der Aalst, Ton Weijters, and Laura Maruster. Workflow mining: Discovering process models from event logs. *IEEE Transactions on Knowledge and Data Engineering*, 16(9):1128–1142, 2004. doi:10.1109/TKDE.2004.47.
- 104 Jussi Vanhatalo, Hagen Völzer, and Jana Koehler. The refined process structure tree. In *Proceedings of the 6th International Conference on Business Process Management (BPM)*, volume 5240 of *LNCS*, pages 100–115. Springer, 2008. doi:10.1007/978-3-540-85758-7_10.
- 105 Zeno Vendler. Verbs and times. *The Philosophical Review*, 66(2):143–160, 1957. doi:10.2307/2182371.
- 106 Raphael Volz, Steffen Staab, and Boris Motik. Incrementally maintaining materializations of ontologies stored in logic databases. *Journal on Data Semantics II*, pages 1–34, 2005. doi:10.1007/978-3-540-30567-5_1.
- 107 Mark D. Wilkinson, Michel Dumontier, IJsbrand Jan Aalbersberg, Gabrielle Appleton, Myles Axton, Arie Baak, Niklas Blomberg, Jan-Willem Boiten, Luiz Bonino da Silva Santos, Philip E. Bourne, et al. The FAIR guiding principles for scientific data management and stewardship. *Scientific Data*, 3(1):1–9, 2016. doi:10.1038/sdata.2016.18.
- 108 Glynn Winskel. *The formal semantics of programming languages - An introduction*. MIT Press, 1993. doi:10.7551/mitpress/3054.001.0001.
- 109 Benjamin Zarriß and Jens Claßen. Verification of knowledge-based programs over description logic actions. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI)*. AAAI Press, 2015. doi:10.25368/2022.216.
- 110 Yue Zhao, Guoyang Chen, Chunhua Liao, and Xipeng Shen. Towards ontology-based program analysis. In *Proceedings of the 30th European Conference on Object-Oriented Programming (ECOOP)*, pages 26:1–26:25, 2016. doi:10.4230/LIPIcs.ECOOP.2016.26.

Grounding Stream Reasoning Research

Pieter Bonte ✉ 🏠 

Department of Computer Science, KU Leuven Campus Kulak, Belgium

Daniel de Leng ✉ 

Linköping University, Sweden

Emanuele Della Valle ✉ 

DEIB - Politecnico di Milano, Italy

Federico Giannini ✉ 

DEIB - Politecnico di Milano, Italy

Konstantin Schekotihin ✉ 

Alpen-Adria-Universität Klagenfurt, Austria


Alessandra Mileo ✉ 🏠 

Insight Centre for Data Analytics, Dublin City University, Ireland

Riccardo Tommasini ✉ 🏠 

INSA Lyon, CNRS LIRIS, France


University of Tartu, Estonia

Giacomo Ziffer ✉ 

DEIB - Politecnico di Milano, Italy

Jean-Paul Calbimonte ✉ 🏠 

University of Applied Sciences and Arts Western Switzerland HES-SO, Sierre, Switzerland

Daniele Dell’Aglio ✉ 🏠 

Aalborg University, Denmark

Thomas Eiter ✉ 

Technische Universität Wien, Austria

Fredrik Heintz ✉ 

Linköping University, Sweden

Danh Le-Phuoc ✉ 🏠 

Technical University Berlin, Germany

Patrik Schneider ✉ 

Technische Universität Wien, Austria

Siemens AG, Chemnitz, Germany

Jacopo Urbani ✉ 

Vrije Universiteit Amsterdam, The Netherlands

Abstract

In the last decade, there has been a growing interest in applying AI technologies to implement complex data analytics over data streams. To this end, researchers in various fields have been organising a yearly event called the “Stream Reasoning Workshop” to share perspectives, challenges, and experiences around this topic.

In this paper, the previous organisers of the workshops and other community members provide a summary of the main research results that have been discussed during the first six editions of the event. These results can be categorised into four main research areas: The first is concerned with the technological challenges related to handling large

data streams. The second area aims at adapting and extending existing semantic technologies to data streams. The third and fourth areas focus on how to implement reasoning techniques, either considering deductive or inductive techniques, to extract new and valuable knowledge from the data in the stream.

This summary is written not only to provide a crystallisation of the field, but also to point out distinctive traits of the stream reasoning community. Moreover, it also provides a foundation for future research by enumerating a list of use cases and open challenges, to stimulate others to join this exciting research area.

2012 ACM Subject Classification Information systems → Data streams; Information systems → Stream management; Information systems → Graph-based database models; Information systems → Query languages for non-relational engines; Computing methodologies → Temporal reasoning; Computing methodologies → Description logics; Information systems → Semantic web description languages

Keywords and phrases Stream Reasoning, Stream Processing, RDF streams, Streaming Linked Data, Continuous query processing, Temporal Logics, High-performance computing, Databases

Digital Object Identifier 10.4230/TGDK.2.1.2

Category Position

Funding J.-P. Calbimonte acknowledges support from the Swiss National Science Foundation under grant No. 213369 (*StreamKG*), and from the EU Horizon Europe program under grant No. 101092908 (*SmartEdge*). D. Le-Phuoc is supported by the Deutsche Forschungsgemeinschaft, German Research



© Pieter Bonte, Jean-Paul Calbimonte, Daniel de Leng, Daniele Dell’Aglio, Emanuele Della Valle, Thomas Eiter, Federico Giannini, Fredrik Heintz, Konstantin Schekotihin, Danh Le-Phuoc, Alessandra Mileo, Patrik Schneider, Riccardo Tommasini, Jacopo Urbani, and Giacomo Ziffer; licensed under Creative Commons License CC-BY 4.0

Transactions on Graph Data and Knowledge, Vol. 2, Issue 1, Article No. 2, pp. 2:1–2:47



Transactions on Graph Data and Knowledge

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Foundation under grant number 453130567 (*COSMO*) and by the Horizon Europe Research and Innovation Actions under grant number 101092908 (*SmartEdge*). A. Mileo acknowledges support from Science Foundation Ireland (SFI) Grant Number SFI/12/RC/2289_P2, co-funded by the European Regional Development Fund. T. Eiter acknowledges support from the Vienna Science and Technology Fund (WWTF) project ICT22-023 (*TAIGER*) and the EU Horizon Europe program under grant No. 820437 (*Humane AI Net*).

Received 2023-09-18 Accepted 2023-11-17 Published 2024-05-03

Editors Aidan Hogan, Ian Horrocks, Andreas Hotho, and Lalana Kagal

Special Issue Trends in Graph Data and Knowledge – Part 2

1 Introduction

Stream Reasoning (SR) has emerged as a branch of artificial intelligence that draws attention to the need to make decisions incrementally, as soon as possible, and before they are no longer helpful. Such an ambitious and broad goal requires many competencies as it entails different research problems. As a result, Stream Reasoning bridges several research communities, such as Knowledge Representation, Robotics, Data Management, and Semantic Web, and it has found applications in various application domains, including traffic management, social media analytics, and robotics.

For more than a decade, the stream reasoning community has proceeded with a shared vision and provided many independent contributions. In this paper, a few community members, some active since the beginning and some recently welcomed provide an overview of the leading research contributions within the Stream Reasoning field discussed during these events. Moreover, this article aims to crystallise the notion of stream reasoning, examining how these different communities contributed to various aspects of its research vision and highlighting the overlaps and peculiarities. The inputs of this crystallisation process were the programs and discussions of the past workshops and the results of a questionnaire prepared specifically for this article. Some authors prepared the questionnaire, starting from one of the initial research questions [75], called **Q** henceforth, that contributed to the fundamental vision of Stream Reasoning:

(**Q**) Can we make sense in near real-time of vast, rapidly evolving, constantly varying, inevitably noisy, incomplete and heterogeneous data streams coming from complex domains?

This question touches upon the various research dimensions related to SR: *Near real-time* pertains to the urgency of obtaining an answer; it is essential to secure a response as swiftly as possible and definitely before the information loses its value, a notion referred to as *velocity*. A unique challenge is given by *heterogeneous data*, emphasising the variety of data, where data is not uniformly formatted. The term *noisy* refers to the inherent uncertainty about and in the data. When one mentions data being *vast*, they point to the immense volume of data generated in a given time frame, signifying scalability challenges. *Incomplete* data suggests an absence of specific data or information in the stream. The description *rapidly evolving* underscores the unpredictable nature of the stream's ingestion rate, while *constantly varying* speaks to the unpredictability of the content of the stream and its potential constituents. In conclusion, the term *Complex domains*, which perhaps distinguishes Stream Reasoning from the related topic of Stream Processing, is reserved for those application areas where merely validating data is not sufficient; these domains necessitate capturing and integrating semantics, complex relations between parts, and context through a more expressive language.

Question **Q** can also be used to define a macro-level perspective on decision-making. In social sciences, macro-level questions correspond to the collective investigation of a research field, i.e., they are used to define the research context [131]. As such, they remain unanswered regardless of individual contributions, and thus, they shall be reduced to simpler lower-level questions. In particular, two additional levels are expected:

- **Meso level:** adds requirements to limit the research context, but still unsolvable. Meso-level questions roughly correspond to the investigation of several PhD theses.
- **Micro level:** reduces the investigation to a measurable outcome that can assess the validity of the contribution, e.g., a research paper.

Therefore, we used the answers to the questionnaire to reformulate **Q** into meso and micro questions to characterise the different areas within Stream Reasoning research. Moreover, we asked the participants to sustain their answers with a thorough analysis of the Stream Reasoning state of the art. Our goal is indeed grounding the pillars of Stream Reasoning research to the extent of guiding the future of this research community. The result redefines the aforementioned terms into four partly overlapping areas of research:

Stream Processing, i.e., the area concerned with developing systems that can efficiently process large data streams. Given the focus on data management, this research area is traditionally embedded in the database and complex event processing communities.

Streaming Linked Data, i.e., the area that focuses on extending the Semantic Web stack to deal with streaming data. Because of this, contributions in this area are primarily presented in the Semantic Web community.

Deductive Stream Reasoning, i.e., the area that focuses on designing deductive reasoning techniques that can infer implicit knowledge from the stream. Most techniques in this area are based on logic-based methods and come from the Knowledge Representation community.

Inductive Stream Reasoning, i.e., the area that studies how we can infer new knowledge using inductive reasoning techniques. To this end, the most recent contributions exploit the latest developments in Machine Learning to learn new knowledge from the data.

While SLD encompasses several areas of research that are interested in data sharing and integration for evolving data, inductive and deductive stream reasoning focus on efficiency and expressiveness. To clarify the difference between deductive and inductive reasoning, in deductive reasoning, one evaluates logical statements to make implicit knowledge explicit; prototypical reasoning is making proofs in a logical calculus, applying rules, etc. For example, from a and $a \rightarrow b$ we may conclude b . Notably, the reasoning is sound. In inductive reasoning, one infers rules from data; e.g., from images showing white swans, one may infer that, as a rule, swans are white. In contrast to deductive reasoning, inductive reasoning is not generally sound, and the result may be incorrect. To address this, inferences may be drawn under uncertainty, often resorting to probabilities. Notably, deductive reasoning may involve uncertainty, but all knowledge is already implicit.

We will discuss each of the areas in the following four sections. The discussion will follow the same structure in each section. First, we will formulate a meso question for that specific sub-area of stream reasoning, and then we will dig into the following sections:

- In the “Make Sense” section, we investigate the standard way to express a Stream Reasoning problem in that sub-area, e.g., continuous querying or logical program.
- In the section “Taming Volume”, we describe what research efforts in that particular sub-area address the scalability problem, e.g., using distributed systems to scale out;
- In the section “Taming Velocity”, we focus on those research efforts in that sub-area that relate to the hurdle of processing data as soon as possible, e.g., adopting window-based processing;

2:4 Grounding Stream Reasoning Research

- In the section “Taming Variety”, we discuss how existing works in that sub-area approach the challenge of information integration, e.g., using graph-based data models;
- In the section “Domain Complexity”, we present the results for representing domain knowledge, e.g., ontologies;
- Finally, in the “Data Quality” section, we discuss what methods were adopted or assumptions were made regarding data quality issues, e.g., missing data.

We will conclude in Section 6 with a discussion on how the various areas relate, primarily pointing out when they overlap and how to move forward in this exciting field with a list of what we view as critical open challenges.

2 Stream Processing

Stream Processing is a technological solution and a research field that first addressed the problem of continuously analysing data in near-real-time. Stream Processing pre-dates Stream Reasoning research. Indeed, Execution models for Stream Processing have been around for decades [159]. Therefore, it had a direct influence on Stream Reasoning research. At the same time, the push towards a broad form of intelligence and decision support that does not neglect reactivity, which is one common theme in Stream Reasoning research, had a return on Stream Processing as a field.

Thus, it makes sense to look at Stream Processing from a Stream Reasoning perspective, to understand how it contributes to the latter vision. To this extent, we formulate the research question below to capture the objectives of Stream Processing research that align with the one captured by the macro question:

Meso (Stream Processing): Can we continuously query, using declarative SQL-like languages, vast, rapidly evolving, constantly varying, potentially noisy data streams, minimising latency and maximising throughput?

In the remainder of the section, we discuss how Stream Processing has answered such a question.

Stream Processing covers the whole life-cycle of streaming data: from their ingress to manipulation and eventual egress.

2.1 Make Sense

As motivated by Cugola and Margara [67] in their overview of what they call information flow processing systems:

Many distributed applications require continuous and timely information processing as they flow from the periphery to the system’s centre.

Such Stream Processing systems are designed to support large applications in which data are generated from multiple sources and pushed asynchronously to servers responsible for analysing them [115]. Traditionally, analytics is the main objective of the processing, with Stream Processing systems focusing on low-latency, high-throughput online analytical processing (OLAP) workloads.

In terms of **making sense** of the data, Stream Processing introduced the notion of Continuous Querying, i.e., queries that continuously run against streaming or real-time data to produce results or output whenever new data meets the query’s conditions. Traditional database queries are one-time operations: a query is executed and results are obtained based on the current state of the database. In contrast, continuous queries persist and constantly check incoming data.

■ **Table 1** Description of the taxi *ride* stream data. ■ **Table 2** Description of the taxi *fare* stream data.

field	description	field	description
<i>rideId</i>	the unique ride id	<i>rideId</i>	the unique ride id
<i>taxiId</i>	the unique id for the taxi itself	<i>taxiId</i>	the unique id for the taxi itself
<i>driverId</i>	the unique id of the taxi driver	<i>driverId</i>	the unique id of the taxi driver
<i>isStart</i>	has the ride has started or ended	<i>startTime</i>	the time the ride started
<i>eventTime</i>	timestamp of the event	<i>paymentType</i>	the type of payment(<i>cash/card</i>)
<i>startLon</i>	the longitude where the ride started	<i>tip</i>	the tip amount for the ride
<i>startLat</i>	the latitude where the ride started	<i>tolls</i>	the amount of tolls paid
<i>endLon</i>	the longitude where the ride ended	<i>totalFare</i>	the total fare
<i>endLat</i>	the latitude where the ride ended		
<i>passengerCnt</i>	the number of passengers		

Continuous Queries are more specialised than general coding tasks, and thus, they are typically supported by algebra or formal semantics. To our knowledge, the first appearance dates back to the seminal work of Terry et al. [211]. Since then, continuous queries have been discussed extensively [23, 16, 62]. Limiting our mention to fully-declarative languages, we can distinguish two families of continuous queries, which differ on the expressivity of the languages they use:

- SQL-Like Languages based on the foundational CQL models by Arasu et al [9]. Such languages allow expression *window-based* continuous queries over relational data streams. Three types of windows have been considered: *time-based (sliding) window* which discards all data beyond a certain point in time; *tuple-based window* which dumps all data that has arrived prior to a predefined number of tuples (e.g., keep only the last 10 facts); *partition-based windows*, which partitions the stream in various substreams based on the attributes of the data in the stream.
- Complex Event Recognition Languages focus on detecting regular expressions over streams of typed events. Although operators like Sequence (follow by) and Allen Algebras are well-accepted, a universally accepted foundational algebra is still missing.

► **Example 1 (Taxi).** To illustrate the difference between the research areas, we provide examples of various queries typical for each research area. We will utilise the taxi dataset provided by the ACM DEBS 2015 Grand Challenge¹ as an ongoing example. The DEBS challenge centers around analysing taxi routes within the city of New York. The dataset encompasses two streams: the *ride* stream, which describes taxi journeys including (i) taxi specifications, (ii) pick-up and drop-off details (such as geographical coordinates and timestamps); and (iii) passenger count; and the *fare* stream, which describes payment details for the rides (such as tip, payment method, and total fare). Specifically, Table 1 outlines the attributes found in the *ride* stream, while Table 2 delineates the attributes in the *fare* stream. Note that *rideId*, *taxiId*, and *driverId* are contained in both streams.

Listing 1 shows an example of a CQL query that combines both streams, counting all the rides over the last hour that had more than 2 passengers and cost more than 10 dollars. The *Istream* operator in the *Select* clause describes that the result of the query will be a new stream containing the new results within the window of 1 hour.

¹ <http://www.debs2015.org/call-grand-challenge.html>

```

1 Select Istream(Count(*))
2 From RideStream [Range 1 Hour Slide 1 Minute]
3 From FareStream [Range 1 Hour Slide 1 Minute]
4 Where RideStream.rideId = FareStream.rideID AND
5     RideStream.passengerCnt > 2 AND
6     FareStream.totalFare > 10

```

■ **Listing 1** An example of an CQL query on the taxi stream.

Carbone et al. [59] studied the field’s maturity concerning processing. Initially, research focused on languages and paradigms for continuous querying and designing Data Stream Management Systems (which extend Data Base Management Systems to support continuous semantics). Later, research moved towards Scalable Stream Processing, motivated by the advent of Big Data challenges. More recently, the authors claim, Stream Processing is moving beyond analytical workloads, welcoming concepts like database transactions, stateful functions, and model serving. Moreover, Stream Processing has been applied beyond continuous queries, addressing tasks such as conformance checking [185], continuous pattern-matching streaming graphs [169, 168], and graph partitioning [1].

2.2 Taming Volume

As highlighted by Carbone et al., the first generation of streaming systems was centred around proving the feasibility of continuous querying and paying little attention to scalability. Hence, the first generation of streaming engines is limited to vertical scaling, e.g., IBM System S, Esper, Oracle CQL/CEP and TIBCO.

Later, due to the introduction of MapReduce and the popularisation of cloud computing, Stream Processing research and development started shifting to the scalability problem. Although velocity (described below) was always the priority, data parallel and distributed solutions became the de facto standard.

In particular, it is worth mentioning Apache Flink [60], which uses a streaming dataflow engine that provides data distribution, communication, and fault tolerance for distributed computations. Apache Flink features two relational APIs – the Table API and SQL—for unified stream and batch processing. Flink’s Streaming SQL support is based on Apache Calcite, which implements the SQL standard. Apache Spark [11] is a versatile distributed computing platform that offers convenient programming interfaces in Java, Scala, Python, and R, along with a well-optimised engine that is capable of handling various execution graphs. At the core of Spark’s abstractions are resilient distributed datasets, which represent collections of elements distributed across nodes within the cluster, enabling parallel data processing. Apache Kafka [231] works as a distributed streaming platform, operating as a cluster on one or more servers called brokers. This cluster can span across multiple data centres. Kafka’s primary role is to store continuous streams of records in what are known as topics, which are essentially unbounded, append-only log structures. Each record within these topics comprises three main components: a key, a value, and a timestamp. A Stream Processing library called Kafka Streams is also built on Apache Kafka’s producer and consumer APIs. It operates on a model known as Stream/Table duality [195].

2.3 Taming Variety

The support for data heterogeneity is limited in general streaming systems. Indeed, Data Stream Management Systems (DSMS) and Complex Event Processing (CEP) engines inherit their data model and query languages from the database community. The seminal work from Babu et Widom [16] poses the basis for relational Stream Processing and influences various languages.

The data models are evolving, with stream processing systems supporting more complex data types inspired by object-oriented programming languages. Indeed, Flink, Spark, Kafka Streams and many more support nested data structures, allowing users to design hierarchies of event types.

Notably, the approach taken from existing DSMSs to address the data variety is rather practical and lacks formal foundations. Data integration is performed through custom data pipelines rather than following information integration principles [140]. Conversely, relational languages have been extended to navigate simple nested structures like JSON. For example, Spark SQL has included operators to manipulate CSV and JSON data since 2017. KSQL and Flink added the opportunity to access nested fields in JSON data within the SQL dialect last year. Nonetheless, data access is managed without source data mapping, making fraternisation somewhat arbitrary and porting queries across systems nearly impossible when semistructured data are involved. Although the notion of an event, as a typed notification of fact at a given time, can be seen as a shared abstraction that can glue DSMS together, few attempts remain in the realm of Stream Processing systems.

It is worth noticing, though, that there are emerging more specialised Stream Processing systems capable of handling more sophisticated data structures such as interval-based events [15], streaming graphs [169], and property graph streams [93].

Orthogonal to the data representation, the Stream Processing literature distinguishes two types of streaming data, i.e., record streams and change data capture. The former indicates positive tuples like sensor network observations, while the latter describes changes within a database (additions and deletions). Although Stream Processing does not typically consider variety in the data model, these two types of streams typically co-exist in the context of streaming systems [195]. Additionally, in the context of system observability, such a dichotomy has evolved into a trichotomy including metrics, logs, and traces, which represents numerical observations, factors or changes, as well as the propagation of information across systems, respectively [199].

2.4 Taming Velocity

Data velocity, i.e., the requirement for processing data as soon as possible and before they are no longer valuable, is the first and foremost priority for Stream Processing research. The velocity challenge has a direct impact on data storage. Indeed, putting data at rest and processing them later is no longer possible, as it would require too much time. In practice, data velocity is treated by operating in memory. Stream Processing Engines, i.e., systems capable of handling data with high throughput and low latency, employ sophisticated mechanisms to reduce the memory footprint without compromising performance.

Their performance is measured alongside two axes, each representing a key performance indicator, i.e., end-to-end latency (the time passed from when a data point enters the system and when it exits as part of the output) and maximum throughput, the amount of data processed within a unit of time, e.g., a second. The two dimensions are in a clear trade-off, pulling the Stream Processing envelope on from two sides, i.e., incremental vs batch computations.

Another substantial change happens in the query model. Queries are no longer issued online but are instead registered and compiled into pipelines, which typically avoid loops for efficiency. As a query can run indefinitely and until explicitly suspended, the result is a stream of answers. The query evaluation occurs upon the arrival of individual data elements or in small (micro) batches. Punctuation mechanisms, i.e., the presence of particular landmarks in the data or the query, are used to progress the execution in a distributed setting. On the data side, punctuation is the minimal informational unit that constitutes a single item in the stream. On the query side, punctuation assumes the role of operators, commonly named windows, that allow the gathering of multiple elements in the stream that should be processed simultaneously. Ultimately, windows can

introduce a delay in different parts of the framework. Modern engines that simultaneously address the velocity and volume challenge may consume millions of events per second, guaranteeing an end-to-end sub-millisecond latency.

2.5 Domain Complexity

In Stream Processing, the complexity of the domain is usually considered relatively limited. Domain modelling is reduced to relational and document data when considering production-graph Stream Processing systems like Flink, Spark Streaming and the Kafka suite. Notably, the presence of a schema, be it relational or document-based, as in the case of binary formats like Avro or JSON Schema, is essential to decouple the production and consumption of streaming data. In terms of conceptual modelling, approaches for event data representation emerged, e.g., event sourcing [35], but only as methods for system integration and without formal semantics [165]. Lastly, in Complex Event Processing, hierarchical data models are often adopted but limited to taxonomical relations inspired by inheritance in object-oriented programming languages [104].

The adoption of Stream Processing systems into application domains that require strong consistency guarantees, e.g., financial analysis or traffic management applications, called for more sophisticated domain modelling techniques. While on the conceptual level, everything remains unchanged, at lower levels of abstraction, the Stream Processing engines require awareness of partitioning schemes, possible faults, and out of order. To this extent, researchers have focused on *consistency* in terms of transactional behaviour [238, 2, 50]. The ACID properties, which are standard in the database context, ensure that the (database) state is consistent to the degree required by a given isolation level. In Stream Processing, the focus shifted to the interaction across systems. Thus, the notion of consistency is discussed in terms of delivery guarantees: *At-least-once* ensures that input data are not lost, *at-most-once* eliminates duplicate processing, and *exactly-once* combines both, ensuring the absence of input data losses and repeated delivery of results [212]. The definition of such guarantees is expressed at the logical level: individual data items are extended with metadata to be used downstream for controlling consumption. Transactional Stream Processing is an ongoing research that is gaining traction at the industrial level².

Last but not least, the role of provenance in Stream Processing represents the most notable attempt to manage additional domain complexity, i.e., reason about the *why* and the *how* of continuous query answers [105]. Vijayakumar et Plale [224] first proposed a low-latency method for generating coarse-grained provenance information that focuses on capturing dependencies between different data streams instead of individual tuples. Wang et al. [232] spot the limitations of techniques based on annotations and suggest a rule-based approach for provenance in Stream Processing applied to the medical domain. However, this approach requires access to all intermediate streams, making it less suitable for modern Stream Processing systems. Glavic et al. [106] proposed a set of instrumented operators to track the provenance of select-project-join queries in Stream Processing scenarios. More recently, the works of Palyvos-Giannas explore richer provenance models, in particular: *Ananke* allows users to track richer provenance information, not only specifying which source tuples contribute to which query results but also whether each source tuple can potentially contribute to future results [171]. *GeneaLog* is similar to *Ananke* but with a focus on the edge [170]. Finally, *Erebus* investigates the aspect of *completeness*, relying on why-provenance [172] for identifying missing answers in the result by explaining the mismatch between actual and expected answers for continuous queries. As such, explaining the inconsistency of continuous queries is not applicable.

² <https://github.com/ververica/streaming-ledger>

2.6 Data Quality

Several factors impact streaming data quality, e.g., noisiness, incompleteness, and timeliness. Stream Processing systems must be able to handle situations where individual data points are missing, entire data streams stop suddenly, or queries are changed. These situations can occur as the result of for example data loss during transmission, changing streaming resources, or changing user/agent needs. This can be regarded as an orchestration problem, where resources are carefully managed to minimise latency and maximise throughput even in the face of changing circumstances.

3 Streaming Linked Data

Throughout the years, the Semantic Web has built and standardised a stack of technologies that enable the vision of publishing, accessing, and processing data on the Web, as if in a database [34]. Among these technologies and standards, IRIs [88] are used to identify resources, RDF provides a data model to describe such resources and their relations in graph-based data structures, ontologies such as RDFS/OWL offer languages to specify schemas (consisting of concepts and the relations between these concepts), and SPARQL provides a declarative query language to execute CRUD operations on RDF graphs (to Create, Read, Update, and Delete resources).

These technologies were built without including time as an intrinsic part of their data model. While it is easy to understand this choice – many systems do not deal with time or delegate its management to the application layer – data evolves, and it is often necessary to address it. Therefore, the community started to build time-aware solutions on top of the Semantic Web stack. For instance, there have been initiatives at the modelling level, such as OWL-Time [65] that allow defining temporal concepts, or the Semantic Sensor Network ontology [64], which provides a vocabulary to describe sensor observations over time. The Semantic Web standards themselves evolved, accounting for time. For example, the RDF 1.1 recommendation [69] states:

The RDF data model is atemporal: RDF graphs are static snapshots of information. [...] RDF graphs can express information about events and temporal aspects of other entities, given appropriate vocabulary terms.

In practice, this implies that time information can be included within an RDF graph, without time-specific semantics. In addition to use cases where it is necessary to account for time, a second need emerged: responsiveness. An increasing number of applications require not only managing temporal data, but also timely processing of results. These requirements are frequent in a large number of domains including social media analytics on the Web, or data management for the Web of Things (WoT). In these applications, it is vital to provide instantaneous query and analysis results, for which time order and recency play a crucial role.

These needs led to a novel research area within the Semantic Web community, under the denominations of *RDF Stream Processing* (RSP) or *Streaming Linked Data* (SLD) [217]. RSP research has focused more on the temporal extensions for RDF data and query modelling, while SLD has centred on the implications of Stream Processing for graphs that comply with the Linked Data principles [41, 46]. Beyond these minor differences, this line of research has delineated an agenda that has studied the following aspects:

- Modelling data streams and complex events using RDF graphs, including syntactic, semantic, and operational implications.
- Extending RDF query languages with streaming data operators.
- Building RDF stream Continuous Query processors, including different reasoning and processing variants.

2:10 Grounding Stream Reasoning Research

- Evaluating and benchmarking Stream Processing engines, including performance, and correctness, among other metrics.
- Interconnecting RDF stream processors through Web interfaces.

This allows us to reformulate the macro-level research question to the following specific meso-level research question for RSP/SLD:

Meso (Streaming Linked Data): Can we evaluate Continuous Queries, expressed as a dialect of the SPARQL language, over RDF streams with limited latency while incorporating domain knowledge through RDFS ontologies?

This and other subsequent research questions have been explored and discussed, many of which converged around the RDF Stream Processing Community Group (RSP CG), within the context of the W3C³. This group served as a central discussion square that led to different formalisation, implementation, and benchmarking initiatives in this area.

RSP/SLD has been successfully used in a variety of use cases, ranging from social media analytics [20], traffic monitoring in Smart Cities [139], large-scale streaming data retrieval in Smart Farming [124], to monitoring the performance of athletes [158] and the health of patients [71].

We will now explain how RSP/SLD research has targeted different aspects of the original Stream Reasoning macro-level research question.

3.1 Make Sense

In order to process RDF streams, it was observed that Semantic Web technologies and Stream Processing technologies are complementary for solving the problems that Stream Reasoning tries to tackle. In terms of **making sense** of the data, RSP and SLD are fundamentally based on *continuous querying* and *data integration* approaches. The former, takes the idea from SP, where queries are registered only once and continuously produce results as they are evaluated over streams of data. Moreover, RSP and SLD inherit the data integration capabilities from the Semantic Web, as they seamlessly integrate Stream Processing and Semantic Web technologies. Through the use of ontology models to represent the stream data elements, these query languages were able to integrate different data sources, including both static and streaming data.

Over the years, several languages have been proposed. Most of them aim to process extensions of RDF where triples or graphs are annotated with temporal information, such as individual timestamps or time intervals. Examples of these languages include C-SPARQL [24], Streaming SPARQL [43], CQELS-QL [134], or SPARQL-Stream [55]. Most of these languages extend the SPARQL syntax with time-based sliding window operators, as found in Stream Processing; and in some cases, additional query functions. The semantics of how these windows work, however, were not uniform and were shown to have different operational behaviours. In consequence, these languages disagreed on the correctness of query results in certain cases [81], as they had different properties that made them difficult to compare,

To address this issue, a unifying formalisation of continuous query processing over RDF streams was proposed in [78], named RSP-QL. This model was able to include streaming query evaluation semantics, as well as operational semantics of windows, thus allowing to characterise existing extensions of SPARQL for continuous querying. The ability to represent different types of queries using RSP-QL is a first step towards the standardisation of continuous querying extensions for

³ W3C RSP Community Group: <https://www.w3.org/community/rsp/>.

RDF streams. However, RSP-QL still inherits practical inconveniences from SPARQL, such as the difficulty of generating and re-consume RDF stream results (e.g., through using the SPARQL CONSTRUCT clause).

There are more operators beyond the sliding window operator. Examples include basic CEP sequence matching, which was integrated into RSP-QL through RSEP-QL [82], and monotonicity conditions, which were proposed in STARQL [166]. Nevertheless, these elements add substantial complexity to the formalisation and eventually to the implementation of querying engines, as we will see next.

3.2 Taming Volume

Although most of the contributions in the SLD research area have focused on addressing the inherent velocity of data streams, taming volume has not been thoroughly investigated. There have been some efforts, such as CQELS-Cloud [135] and Strider [189] that build upon the elasticity of existing Stream Processing frameworks, respectively Apache Storm and Apache Spark. However, the focus of taming huge volumes of data has been rather limited.

Nevertheless, this dimension has indirectly been addressed through the analysis of query execution efficiency and response time constraints. Velocity can be analysed in terms of volume over time, which was analysed in RSP benchmarking efforts [239, 133]. Among the works specifically targeting stream data volume we can mention efforts for reducing the actual size of serialised RDF streams, using the compressed ERI interchange format [94]. The usage of reduced formats for RDF stream data exchange are of primary importance for IoT environments where message volume is critical [122], such as constrained devices, limited network bandwidth, and reduced storage sensors.

Other approaches addressed volume from the processing perspective, for instance proposing load-shedding techniques to limit the number of stream data items to be processed [33], or data eviction strategies to reduce the cardinality in join operations over RDF streams [100].

3.3 Taming Variety

RDF graphs allow modelling all sorts of information on the Web, enabling wide exchange and interoperability. However, these graphs are atemporal, and RSP needs an adequate data model to publish and exchange data streams while handling data variety. *RDF streams* address this challenge by extending the RDF model with notions of time-based order. The initial attempts to define RDF streams were crystallised by the RSP CG, which proposed the following requirements for the abstract model of RDF streams [7]:

- R1** It should be possible to represent RDF streams with an abstract RDF-based model, whose semantics should provide the basis for producing and consuming streams.
- R2** It should be possible to identify an RDF stream using IRIs.
- R3** It should be possible to serialise the RDF stream abstract model into RDF formats derived from existing standards, extending them only when necessary.
- R4** It should be possible for RDF Stream to have timestamps based on different notions of time (time instants, intervals) with different semantics (application, validity, transactional).
- R5** In case no timestamp is associated with an RDF stream data element, the system should be responsible for managing the time-based ordering of stream elements.
- R6** It should be possible to restrict the RDF stream model to facilitate implementation and support efficient representation.

```

1 prefixes:
2 taxi: "http://linkeddata.stream/ontologies/taxi#"
3
4 mappings:
5 rides:
6   s: taxi:ride/${rideId}
7   po:
8     - [a, taxi:RideEvent]
9     - [taxi:hasEndLon, ${endLon}]
10    - [taxi:hasEndLat, ${endLat}]
11    ...

```

■ **Listing 2** An example RML Mapping on the taxi dataset.

The above requirements call for reusing existing Semantic Web technologies and account for different types of temporal information. One can consider an RDF stream as a sequence of RDF graphs, each identified by an IRI and optionally associated with temporal annotations, such as creation time or validity interval. The generality of the RDF streams definition aims at opening the door to different kinds of streams, which may occur in different application scenarios. However, it also implies the challenge of dealing with the complexity of covering modelling variations. This specification led to the implementation of systems capable of producing streams of RDF data [24, 136, 216]. At the simplest level, plain RDF can be used to represent streaming information without specific semantics for time annotations. For example, the Linked Sensor Data [175] initiative proposed the publication of meteorological sensor data using stored RDF. Although these RDF graphs represent observations that were originally streamed by sensors, with explicitly recorded time annotations, the system only provided static access to the data. RDF libraries such as Jena⁴, RDF4J⁵, or RDFLib⁶ provide IO methods to read and write plain RDF graphs in a streaming fashion, but they do not support producing and consuming RDF streams.

TripleWave [150] is one of the systems that addressed this limitation, proposing a full pipeline for the generation of RDF streams. It included the production of live RDF streams consisting of time-annotated graphs, which could be fed from non-RDF data sources.

More recently, RMLStreamer was introduced, focusing on the generation of RDF streams in a low latency and high throughput fashion. RMLStreamer is a parallel and scalable Stream Processing engine built on Apache Flink that is able to generate RDF streams from heterogeneous data streams of any format (e.g., JSON, CSV, XML, etc.), using RML mappings [83].

► **Example 2** (Taxi cont'd). SLD allows to integrate the taxi streams with additional static information, e.g., a dataset that describes Points of Interest (POI) within the city. The use of SLD enables this integration, even though the underlying data representation of the POI dataset is not compatible with the raw taxi streams. Mapping the taxi streams and POI dataset to RDF allows to smoothly integrate both datasets, allowing to make more informed decisions regarding the available data. The taxi streams can be mapped to RDF in a streaming fashion through the RMLStreamer. Listing 2 shows how this mapping can be defined in YARRML [114], i.e., a more concise RML syntax. The mapping defines how various fields of the taxi dataset, e.g., *rideId*, *endLon* and *endLat*, can be converted to RDF triples. A similar mapping can be conducted for the POI dataset, regardless of its underlying data format.

The example RSP-QL query in Listing 3 counts all taxi drop offs near a hospital in the last hour, which has been made possible through the integration of the POI dataset. Joins in RSP-QL can be applied to a combination of both windows and stored graphs as seen in the example.

⁴ <https://jena.apache.org/>

⁵ <https://rdf4j.org/>

⁶ <https://rdflib.dev/>

RSP and SLD are thus able to tame data stream variety by reusing and extending Semantic Web technologies, by introducing fundamental temporal semantics into the data model, and by providing tools that implement them.

3.4 Taming Velocity

Each query language designed to process RDF streams was accompanied by working prototypes, also called RSP engines. The first contributions investigated how Semantic Web technologies can be combined with DSMS and CEP engines in order to incorporate Stream Processing capabilities that could target velocity. These systems propose different approaches to continuous query answering over data streams, shifting from the query-response paradigm of conventional SPARQL engines. In addition, these systems have a special focus on reactive question answering, proposing methods that allow for delivering results as soon as possible. Each of these systems incorporates variations of windowing implementations, in order to limit the possibly unbounded stream in processable chunks.

Among the first generation of RSP engines, C-SPARQL [24] adapts a black box approach by pipelining a DSMS with a SPARQL engine. The DSMS is used for handling the Stream Processing capabilities of the engine, e.g., windowing the stream into processable chunks. Each window is then fed to the SPARQL engine for evaluation of the query. In contrast, the CQELS engine [134] employs a white box approach; instead of pipelining existing systems, it integrates the Stream Processing operators in the evaluation of the SPARQL query, opening up various opportunities for optimisation. Morph-streams [56] takes a different approach and uses Ontology Based Data Access (OBDA, or Virtual Knowledge Graphs) to virtually process RDF streams, while their underlying representation is still the raw data (e.g., a relational data stream, or a streaming CSV). It uses a mapping language, i.e., R2RML⁷, to define the relation between the underlying relational data and RDF. Morph-streams uses query rewriting to virtually answer SPARQL-like queries over relational data streams, giving the illusion data is available in RDF.

Other approaches focused on extending existing infrastructures for distributed data processing, such as the aforementioned Strider (see Section 3.2). Regarding the integration and interoperability of RSP engines, RSP4J [216] proposed an API for the development of RSP engines under RSP-QL semantics, providing many of the needed abstractions and interfaces that can be used for building blocks when creating new RSP engines or testing out algorithms and optimisations. RSP4J

⁷ <https://www.w3.org/TR/r2rml/>

```

1 PREFIX taxi: <http://linkeddata.stream/ontologies/taxi#>
2 PREFIX : <http://linkeddata.stream/resource/>
3 SELECT (COUNT(?d) AS ?num_hospitalDropOff)
4 FROM NAMED <citymap.rdf>
5 FROM NAMED WINDOW <w> ON :taxiStream [RANGE PT1H STEP PT5M]
6 WHERE {
7     Graph <citymap.rdf> {?place :hasLat ?lat; :hasLon ?lon; :hasPOI ?poi.
8         ?poi a Hospital. }
9     WINDOW <w> { ?d a taxi:DropOffEvent; taxi:hasEndLon ?lon; taxi:hasEndLat ?lat. }
10 }

```

■ **Listing 3** An example of an RSP-QL query on the taxi stream.

also provides two implementations, Yasper and CSPARQL2.0, that follow the RSP4J interfaces. Following the principles of RSP4J, but written in Rust, RoXi [44] brings RSP engines to the browser through WebAssembly support.

There have been a number of contributions centred on the evaluation of RSP engines. As for Stream Processing solutions, the principal metrics are latency (the time required to process a stream) and throughput (the amount of data processed in a given amount of time). Further metrics include memory footprints, expressiveness (which query operators are supported), and correctness (compliance of a system to its evaluation semantics). Among the proposed benchmarks, LSBench [239] and SRBench [133] proposed re-playable data streams, evaluation queries and a set of metrics to assess the performance and expressiveness of the engines. The YABench [128] framework proposed a more comprehensive coverage of RSP features, while Citybench [4] proposed more realistic and configurable testing datasets. Finally, RSPLab [219] focused on the provision of an open-source environment for RSP reproducibility.

3.5 Domain Complexity

In RSP and SLD, the incorporation of complex domain modelling is usually satisfied by using RDF and RDFS ontologies. In general, the domain complexity in RSP is kept low in order to realise highly reactive systems, given the potential latency that reasoning can add to the query processing stack. Nevertheless, there are some hybrid approaches where RSP and reasoning overlap, for instance, incorporating query rewriting through ontology-based data access or materialising window content and enabling Datalog reasoning. Some of these hybrid approaches are further described in Section 6. When increased domain complexity and expressivity are needed, a sacrifice in latency and throughput is acceptable. To the opposite extreme of this trade-off, we enter the realm of Deductive Stream Reasoning (Section 4), which privileges domain complexity in a dynamic environment.

3.6 Data Quality

Handling *veracity* and *incompleteness* has so far not received much attention within RSP, given that in many cases, the RDF streams are previously pre-processed or fed through streaming pipelines that already perform minimal data cleansing (e.g., through Kafka pipelines [126]). Otherwise, stream data quality control is seldom incorporated into RSP engines. In some cases, Continuous Queries filter out anomalous data, or external data mining and outlier detection modules are employed before the RSP engine receives the stream. When dealing with *constantly-varying data*, Strider and CQELS provide optimisations to reorder the execution of their query execution plans based on the rates of the various streams that are being processed. When the rates of the streams change, the execution plans are reordered to maintain reactivity. The quality of Continuous Query results may sometimes degrade when the stream rate rises. In consequence, it can be helpful in use load-shedding and similar techniques to limit the number of stream items to be consumed [33].

Finally, quality can also be considered regarding the correctness of the Continuous Query processor. In the case of RSP engines, this topic was addressed in [81], which verified that seemingly similar queries resulted in different answers, in some cases not entirely predictable. Based on these results, the operational semantics of RSP query languages have been further studied [78], and other benchmarking frameworks have adopted correctness criteria for their test suites [219].

4 Deductive Stream Reasoning

The contributions presented in the previous sections assume that all the knowledge is stated *explicitly* in the data streams. In some cases, however, there is a wealth of *implicit* knowledge that can be *inferred* with some non-trivial computation.

We refer to systems that do this by evaluating logic-based statements in a deductive manner as *Deductive Stream Reasoners (DSRs)*. Inductive Stream Reasoning, which aims at inferring new knowledge from data, will be considered in the next section.

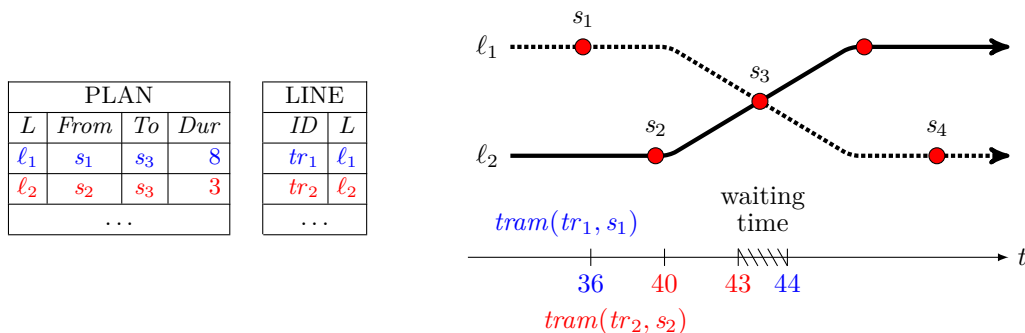


Figure 1 Transportation example.

► **Example 3** (Vienna tram connection). Staying in the transport sector, let us exemplify the general notions of DSRs on the following navigation problem, this time considering public transport instead of taxis: Samantha is travelling in Vienna with her baby and a stroller on a tram ℓ_2 from s_2 to s_4 , which is served by the line ℓ_1 . Thus, Samantha needs to change the line on the stop s_3 .

According to the plan, shown in Figure 1, a vehicle tr_1 that serves the line ℓ_2 requires 3 minutes to reach s_3 from s_2 and a vehicle tr_1 needs 8 minutes to get from s_1 to s_3 . A transportation application that Samantha is using must solve at least the following two problems: (i) get information about the current schedule and delays of trams running on ℓ_1 and (ii) find expected good connections between s_2 to s_4 with less than 5 minutes waiting time at s_3 .

An application based on a DSR gets its knowledge about the transportation problem explicitly, i.e., an expert provides it as a knowledge base KB , such as an ontology or a logic program. DSR then uses KB to solve various problems, e.g., to find suitable routes or inform users about expected arrivals. A data stream comprising information about the current state of the transportation system is pushed to the DSR from sensors and other systems. DSR systems can represent these streams in two possible ways: point-wise or interval-wise. In the *point-wise* representation DSR discretises the time into a set of time point, e.g., a second or a minute, depending on the system architecture. The encoding of data might also vary. Thus, many rule-based DSR systems require incoming data to be encoded as facts, which are associated with time points when they were received, e.g., $36 \rightarrow \{tram(tr_1, s_1)\}$. Other popular representations include database tables, RDF triples, and labelled values in a similar way as the atoms above. The *interval-wise* representation appears to be more natural since time discretisation is not required as in the point-wise case. Hence, a DSR system might associate a set of intervals with every data value appearing in the stream. The main caveat of this representation is that it requires DSR systems to determine the end of each interval. Thus, if a movement sensor reports only changes in tram velocity between the stations, the system cannot determine if the tram is still moving at a constant speed or if the sensor is malfunctioning.

Given the background knowledge about timetables and a stream comprising facts about the positions of trams on their lines, the application needs to retrieve data relevant to Samantha's situation. Most DSR systems use various kinds of *window functions* to retrieve relevant parts of the stream, similar to the windowing introduced in Stream Processing. In our transportation example, a time-based window for the interval $[35, 45]$ will return $\{tram(tr_1, s_1), tram(tr_2, s_2)\}$, and a tuple-based window of size 1 from $t = 45$ will return only $\{tram(tr_2, s_2)\}$.

In general, DSRs are useful in complex domains where applications should be able to continuously make decisions using knowledge explicitly provided by experts. That is, in contrast to the inductive systems, it is not realistic to expect that knowledge required for decision-making is provided in the stream data, like observations and/or labels. Such domains include Cyber-Physical Systems (CPS), Digitalization of Industry, Internet of Things (IoT), and Social Networks. Examples of such applications are

- *monitoring and surveillance*, e.g., of gas turbines [52], maritime vessels [194], or healthcare [111];
- *decision making*, e.g., for video streaming and games [28, 6];
- *planning*, e.g., trajectory planning for UAVs [112];
- *analysis and query answering*, e.g., in social networks [25], smart infrastructures [178], and in intelligent transportation systems [89, 197].

The macro-level SR research question can be reformulated to the following generic meso question for DSR:

Meso (Deductive Stream Reasoning): How can we make knowledge about complex domains, represented in expressive Knowledge Representation languages, available for Stream Reasoning in the realm of vast, rapidly evolving, constantly varying, inevitably noisy, incomplete and heterogeneous data streams?

This generic question gives rise to several concrete meso-level questions that need attention:

1. How can we reuse previously inferred knowledge to minimise the reasoning time upon receiving updates respectively changes in data?
2. How can we extend existing Knowledge Representation languages suitably with temporal operators?
3. How can we achieve a balance between the expressiveness of the Knowledge Representation and the efficiency of reasoning in particular for maintaining a high throughput?
4. How to deal with noise and uncertainty appropriately in expressive Knowledge Representation formalisms, both regarding the quality of results and performance?

Solutions to these questions will be instrumental for achieving the macro goal of Stream Reasoning from above, as rich Knowledge Representation formalisms allow us to express and reason about properties and relationships between data at a deeper level. They enable us to obtain more insight transparently, and provide a basis for developing explanation and justification facilities that will aid in analytics and increase transparency, and hence, trustworthiness.

The first question is at the heart of Stream Reasoning, and requires to face the challenge that conclusions may be obtained by reasoning processes that involve several steps of inferences, depending on the complexity of the underlying Knowledge Representation. *Materialisation*, i.e., computing and storing the valuation of predicates/relations that are defined from given data, and related techniques play an essential role here [156, 225, 157, 221, 177, 118, 222]. *Data parallelisation*, i.e., enabling parallelism in reasoning by data partitioning, has also been considered and investigated as a possible way to tackle this issue [180, 179]. However, for expressive Knowledge Representation languages, incremental evaluation under frequently changing data is not at a level of performance as one would desire in real-time applications like traffic monitoring.

The second question has led to several works in which (static) Knowledge Representation formalisms have been extended with operators and constructs from temporal logics and reasoning, e.g., [86, 112, 52, 29, 226, 58]. However, they are quite diverse and it is at this point open whether the requirements of Stream Reasoning are well covered and which selection and combination of operators would be beneficial, and whether novel operators should be introduced.

The third question is important as, intuitively, constructs in a language that allows for expressing more involving relationships (e.g., joins with negation, nested relations, or recursion) require more computational resources for evaluation [102]. However, even comparatively high resources may not empower one to compute answers over varying inputs, as well-known from descriptive complexity theory [121] and researched extensively in databases and knowledge representation. In DSR, this question—in particular with an eye on high throughput—has been not been much explored yet.

The fourth question arises as commonly declarative languages based on logic assume a well-behaved environment in which data is consistent and uncertainty, if at all, is limited to missing or indefinite information. In the streaming context, this calls for extensions of DSR formalisms that can deal with inconsistent data, outliers, and quantitative uncertainty, especially with probabilistic information. This has been addressed in several works, e.g., [161, 214, 51, 90, 181, 74, 215], but there is no gence nor uniform approach to serve this need, and performance guarantees are an issue. In general, blending uncertainty with logic is a popular topic of interest in AI, with many ongoing works in several communities. In a streaming context, we identify two main research avenues. The first is studying whether existing techniques, in particular those that use deep learning architectures in static contexts, can be successfully adapted so that they can work in a streaming context. The second avenue consists of designing novel techniques specifically for streaming scenarios. This type of combination will be discussed in more detail in Section 5 dedicated on inductive Stream Reasoning.

We conclude by emphasising the critical importance of establishing comprehensive metrics and clear evaluation criteria for assessing contributions to the aforementioned research questions. This is a problem that has been receiving considerable attention in the community (see, e.g., [196]), especially for the following reasons:

- If two solutions implement two different formalisms, then it can be that it is precisely the differences between the two which are responsible for a certain increase/decrease of performance. Thus, it is hard to distinguish the value of a certain solution;
- If we adopt absolute metrics, like runtime or memory consumption, then it becomes arguable when a solution is “good enough” since small variations in the use case can lead to a completely different outcome;
- It is also difficult (or even impossible) to determine which are the most important without resorting to concrete use cases.

As previously mentioned, several RSP benchmarking efforts were developed [239, 133]. These platforms require a graph-based data model and are tailored towards benchmarking query answering, hence are well suited for OWL-based languages. For instance in [57], the authors showed that queries with an OWL2 QL-based engine could be answered up to a throughput of 200K triples/s. Since the mentioned efforts do not cover more challenging reasoning tasks and program sizes, some researchers rely on artificial micro-benchmarks to conduct the experiments and to report empirical evaluations. For instance in [27], LARS-based implementations were compared among themselves and against RSP engines featuring that a response time below 100ms can be achieved for multi-rule programs with a throughput of 800 triples/s. It is likely that a more widespread adoption of DSRs in the real world will guide the research community in choosing more meaningful evaluation criteria.

4.1 Make Sense

A central problem for DSR systems is to promptly answer the question “*What is true now?*”, which has been a widely-studied problem in Knowledge Representation since the inception of the field [96]. The number of contributions made in this area is so high that it is not possible to present a concise summary without running the risk of missing out on some important work. Therefore, we will limit ourselves to pointing the reader to some encyclopedic texts [153, 95, 85, 116, 96] and focus instead on the most recent works that are closely connected to the ones in the other sections.

First of all, let us define a DSR as a system that receives as input a data stream and possibly some background knowledge, either in the form of facts or more complex expressions like rules. The system aims to process the data stream to infer new conclusions using a deductive logic-based process. The computation is specified in a *declarative* manner, that is, we tell the system *what* to compute and let it decide *how to do it*. Typically, it is expected to yield the answers to a given query. For instance, a DSR may receive as input a query in the form of a set of rules and use those to compute the answers.

Since the deductive process is based on logic, DSRs require that the input (stream, query, background knowledge, etc.) is expressed with a formal language. Different such languages have been proposed, based on temporal logic as in the DyKnow framework [112], on extensions of description logics as in SPARQLstream [57] and STARQL [167], or on logical rules as in the popular LARS [29] and DatalogMTL [52, 227] formalisms. The first is grounded on Answer Set Programming (ASP) [53], one of the most well-known languages in the Knowledge Representation community while the other is grounded on Datalog [61], another established formalism in the community. These two languages define the semantics (*what does it mean to answer a query?*) and the supportive expressive power (*what kind of queries can we write?*) in a formal and unambiguous way. In general, we observe here a trade-off that is common with logic-based reasoning: the higher the expressive power is, the more challenging the computation becomes, to the point it is no longer feasible. This trade-off has motivated the design of formalisms, like LARS and DatalogMTL, that have computational bounds that meet the demands of streaming scenarios.

4.2 Taming Volume

First of all, it is essential to mention that while some approaches assume that the stream is infinite (e.g., DatalogMTL [227]), others (e.g., LARS [29]) assume that there is a time point in the future when the stream ends, respectively data beyond it will be ignored. Of course, from a practical point of view, we can set the time when the stream ends to a point which is very far in the future to simulate the case of an infinite stream. From a more formal point of view, however, the assumption that the stream is finite has essential consequences related to the decidability of some critical operations like query answering.

In this context, a *data stream* is often viewed as an ordered collection of timestamped *facts*, e.g., in Example 3 it consists of $tram(36, tr_1, s_1)$ and $tram(40, tr_2, s_2)$. The facts become available as time passes by, which means that the system does not have immediate access to all the data. The data stream is augmented with timestamped atoms that are derived, which in Example 3 may be $exp(44, tr_1, s_3)$ and $exp(43, tr_2, s_3)$ for the projection of the expected arrival times of tram tr_1 and tr_2 , respectively, at stop s_3 . Since we are often interested in obtaining answers immediately, the system must continuously re-evaluate the input queries as new data becomes available. Clearly, to support large volumes of data, it is more efficient to reuse all the inferences previously derived instead of re-computing them from scratch. In a static setting, this problem has been widely studied and is commonly referred to as “incremental reasoning” or “knowledge base maintenance”. Indeed, some of the techniques used for incremental reasoning can be adapted to work on data

streams. For instance, a well-known technique developed for Datalog is *semi-naïve evaluation* [21], which prevents a rule instantiation being evaluated more than once. This technique has been adapted, with some modifications, to work with data streams [27]. Other techniques include multi-shot solving [164], overgrounding of rules [119], and truth-maintenance methods [30] for ASP based stream reasoners.

4.3 Taming Variety

Streams may originate from various sources, such as sensors with different modalities, but also as output of processing components in a system. This naturally leads to a variety of data formats that would need to be accommodated. However, DSR has so far not put much emphasis on heterogeneous data streams, and the systems and approaches available focus on a specific data format. Specifically, as mentioned above symbolic streams are commonly represented as collections of ground atoms that represent any input; the proper treatment and reconstruction of the meaning of the data lies with the stream queries using them. While plain, this approach akin to data models in relational databases still allows for embracing a number of data domains. In some cases like e.g., for RDF, mapping data into logical atoms while preserving the meaning is rather easy, while for richer data formats, such as (part of) a knowledge graph or graph data generated by a camera describing a scene and how it is evolving may be more demanding; *flattening*, i.e., converting structured to plain relation data may serve here as a key technique and predefined data schemes of fixed structure can be used to ease the meaning reconstruction for query answering.

4.4 Taming Velocity

In order to provide responses that are still valuable and not outdated, limiting the data to snapshots is a common approach, in which merely data available at some specific time point is considered. By doing so, one is taking into account that the answer may possibly diverge from the one when the evaluation would happen over the whole stream. *Windowing* is an essential notion in this context, shared among the various approaches, which can be defined as input or dynamically computed. In the first case, the user decides for how long in the past (or in the future) the system is allowed to consider input data. This can be done through *time-, tuple- or partition-based windows*, similar to the windowing functionality in Stream Processing and SLD. In the second case, a reasoner may infer automatically when some data in the past (or in the future) should be ignored. An important aspect in both cases is whether forgetting respectively ignoring data will affect the reasoning outcome; clearly one desires (or may even request) that this is not the case. Unfortunately, the deductive setting comes with computational obstacles: for temporal Datalog, which is a core rule language for temporal reasoning, it is in general undecidable whether forgetting data using finite sliding windows is possible without loss of inferences, as well as to recognize suitable sizes of such windows [191]. Thus, either a (deliberate) loss of inferences is accepted or restrictions on the programs and/or assumptions on the data have to be adopted. In frameworks like LARS, windows can be nested, which seems to occur less frequently in practice. In addition, time points may be abstracted in a window, such that data occurrence *somewhere* (i.e., at some specific point in time) or, dually, *everywhere* (i.e., at all time points) in the window is considered; e.g., DatalogMTL [52, 227] and LARS [29] offer this feature. The language of the i-dlv-sr stream reasoner [58], which leverages on Apache Flink and the incremental ASP solver i²-dlv [119], supports moreover non-contiguous windows that may be time- or tuple-based; however, the rules of a program must use stratified negation, i.e., negation can be evaluated in a layered fashion.

4.5 Domain Complexity

This research dimension is inherently tied to a selected domain language family and the reasoning task at hand, which leads to a wide range of approaches, mainly covered by the fields of Semantic Web technologies and Logic Programming, which adhere to different views of how the world of interest is modelled; this in particular concerns incompleteness of data, which will be addressed in Section 4.6 below.

Temporal Logic. As stream reasoning involves time, temporal logic is a natural basis for DSR. Linear time logic (LTL) [182] is perhaps the most prominent temporal logic. Besides Boolean connectives, temporal operators are available that allow for expressing statements $\mathbf{X}\phi$ and $\phi\mathbf{U}\psi$, which informally mean that ϕ holds in the next stage resp. that ϕ holds always until ψ holds at some stage; $\mathbf{F}\phi$ and $\mathbf{G}\phi$ are shortcuts where $\phi = \top$ (truth) and $\psi = \perp$ (falsity). respectively, meaning that ψ holds at some stage resp. that ϕ always holds. Formulas are evaluated over infinite paths $s_0, s_1, \dots, s_i, \dots$ in a Kripke structure, which intuitively is a transition graph over truth assignments to a set of propositional atoms; this provides a natural link to (infinite) streams. For example, the formula $\phi_1 = \mathbf{G}g \rightarrow \mathbf{F}r$ intuitively expresses that whenever a request (r) is made, it will be granted (g) instantly or at some later stage, while $\phi_2 = \mathbf{G}g \rightarrow \mathbf{X}r$ expresses that whenever a request (r) is made, it will be granted (g) in the next stage; the formula $\phi_3 = \neg g \mathbf{U}r$ intuitively says that no grant occurs prior to the first request. On the infinite path $\emptyset, \emptyset, \{r\}, \emptyset, \{g\}, \{r\}, \{g\}, \emptyset^\omega$, where each set are the atoms assigned true at the respective state, formula ϕ_1 evaluates to true, while ϕ_2 evaluates to false: r is true at stage 2, while g is false at stage 3. The formula ϕ_3 evaluates to true on this path, since r is true at stage 2 and g is false at stages 0 and 1.

Beyond a simple ordinal timeline of consecutive stages $0, 1, 2, \dots$, metric temporal logic (MTL) [130] and variants are considered in DSR in which \mathbf{G} and \mathbf{F} are relative to an interval $I = [a, b]$, written \boxplus_I resp. \boxminus_I , such that $\boxplus_I\phi$ (resp. \boxminus_I) is true at time t in a path, if ϕ is true at every (some) time t' where $t+a \leq t' \leq t+b$. This in particular allows for modelling data snapshots respectively windows as described above, where only part of the stream data is considered for evaluation.

Relational Domains. In a plain relational setting, a domain is similar as with relational databases more or less given by a list of elementary predicates, and any relationships among them have to be expressed by statements in the program or theory for Stream Reasoning.

MTL is used for Stream Reasoning in planning and execution monitoring is [86], which is part of the DyKnow framework [112]. The latter streams data to a monitor which continuously evaluates formulas over them. E.g.,

$$\boxplus((\neg onroad(car1) \vee slow(car1)) \rightarrow \boxminus_{[0,30]}(\boxplus_{[0,10]}onroad(car1) \wedge travel_speed(car1)))$$

may express that if $car1$ is off-road or at slow speed, it will within 30 secs be for at least 10 secs on the road at travel speed. The stream is incrementally incorporated into the formulas by means of the *progression* syntactic rewriting process [17]; this also enables runtime verification.

Rule-based languages can be used to capture complex domains where we distinguish between Prolog-, Datalog-, and ASP-based languages that share rules of the form:

$$a_0 \leftarrow a_1, \dots, a_n, not\ a_{n+1}, \dots, not\ a_m$$

where the a_i 's are first-order atoms and *not* is negation-as-failure (aka default negation).

Pure Prolog was used for implementing real-time complex event detection, such as shown in ETALIS [8] and RTEC [14], as one of its strengths is efficient list processing. Prolog was also more tailored for “native” Stream Processing by lazy evaluation techniques [173] and stream

transducers [174]. The first “streamed Datalog” language is Streamlog [236], which uses the notion of progressive closing world assumption (PCWA) to deal with stream data, considered in Section 4.6 below. Recursive queries further extended the initial language and aggregates in [70]. A different approach was pursued with DatalogMTL extensions [52, 226, 227] allowing for MTL operators in rules that are evaluated over a dense timeline. DatalogMTL was subsequently extended with stratified negation in rules [66] and recently with stable semantics for unstratified rules [229, 228].

Answer Set Programming is well-suited for reasoning tasks that require to model and solve NP-hard search problems. ASP evolved for Stream Reasoning on the level of modelling/language features with StreamRule [152], C-ASP [178] and LARS [29], where all languages introduce various window operators and the latter lifted answer sets to answer streams inheriting their properties (e.g., minimality) and allowing for LTL operators (without next nor until) to be evaluated over a window. LARS was later extended to model quantitative extensions in stream reasoning [90], while StreamRule was later extended to cater for uncertainty [162, 163]. On the level of processing features, the multi-shot solving feature of Clingo facilitated the continuous evaluation of changing logic programs [101]. Fragments of the LARS language were implemented in Ticker [30] and Laser [27], where the later is geared for high throughput on large data volumes with the restriction to stratified programs. Distributed evaluation of LARS programs was introduced in [92], where a program can be decomposed and evaluated by several engines using an interval-based semantics.

Ontologies. Different from the rule-based formalisms above, other DSR languages leverage an ontology of the domain that is given in a customary language, such as the RDF(S) and the OWL2 standard. A basis for temporal reasoning in Description Logics (DL), on which several languages of OWL2 are based, was given in [12], which extended DL with LTL, allowing for temporal operators in DL axioms, with two-sorted semantics for objects and the temporal domain. Furthermore, temporal query answering was investigated, e.g., in [13, 49], where query rewriting over DL-Lite ontologies was extended for LTL operators in queries. A direct extension for RDF(S) ontologies as used in RDF Stream Processing are OWL-based ontology languages such as OWL2 RL, OWL2 QL or OWL2 DL [110]. In particular, OWL2 QL is well-suited for Stream Reasoning since it is first-order-rewritable and can be evaluated on a streaming database system (DBS), i.e., a data management system geared to store and process an incoming data stream in real time. SPARQLstream [57], STARQL framework [167], and the work of [89] allow for query rewriting over a streaming DBS, where the ontology is rewritten into the query that supports window operators. OWL2 RL reasoning that comprises also recursive rules is supported by RDFox [160], where the combination of a main-memory DBS and incremental update enable the use in a stream reasoning setting. An approach that supports more expressive ontologies is TrOWL [213] where the combination of incremental reasoning and with semantic and syntactic approximation of OWL2-DL by OWL2-QL and of OWL2 by OWL2-EL allows for query answering and classification, respectively, over streams of ontologies.

Stochastic Domains. In real-world domains, dealing with quantified uncertainty is an important aspect, which has been addressed in several extensions of DSR languages. PrASP [161] is a probabilistic extension of ASP, which offers probabilistic annotations of formulas, including facts and rules, which induce a possible worlds semantics. LARS has been extended to model quantitative extensions in stream reasoning [90]. Among them is probabilistic reasoning, which has been demonstrated for object tracking in [181]. P-MTL [214] and ProbSTL [215] are probabilistic extensions of MTL and STL respectively, which allow for incremental runtime verification with explicit constraints over deterministic observations and uncertain predictions inside the logic itself. Notably, ProbSTL can express confidence in predictive capabilities by comparing past predictions of the present state with estimations of the current state.

In conclusion, various domain complexities are supported in Deductive Stream Reasoning, ranging from low language complexity as with OWL2 QL and positive Datalog programs, to ASP-related languages, DatalogMTL, and TrOWL, which offer the highest degree of language expressiveness on the level of program structure, temporal operators, and ontology model, respectively. Furthermore, some support of probabilistic reasoning is available.

4.6 Data Quality

Veracity. In approaches based on a crisp 2-valued logical semantics, the data in a stream is expected to be pre-processed and is assumed to be verified in a credible manner. However, facts might still be checked for inconsistencies with respect to the domain knowledge using constraints in rule-based languages, or disjointness in ontology-based languages. In [51], the authors suggested temporal query answering in OWL2 QL over inconsistent data streams with three inconsistency-tolerant semantics designed to automatically repair inconsistencies, e.g., a brave semantics in respect to rigid concepts/roles. If veracity is caused by a sense-reasoning gap between a lower-level probabilistic inference and a higher-level logical reasoning, probabilistic reasoning methods are applied to bridge this gap [113]. Notably, the DyKnow extensions [86] of P-MTL [214] and ProbSTL [215] allow Stream Reasoning with probabilistic temporal logics, where complex formulas can be embedded in probability conditions such as $Pr(a \leftarrow b) < 1$. In [181], the authors followed a different approach by designing a neuro-symbolic stream fusion framework, which includes the learning of rule weights that are mutually independent probabilities. The LARS language was generalised to quantitative extensions in [90] using weighted logic over semirings, which are algebraic structures with multiplication and addition, e.g., the natural numbers, the integers, etc., obeying specific reasonable axioms. Weighted LARS allows lifting several quantitative extensions of logic programming to the streaming setting, among them Problog [184], P-Log [22], and LP^{MLN} [233]. In particular, weighted rules of the form $0.8 : a \leftarrow b$ are supported that induce a probability distribution over the possible answer sets (models) of a program.

Incompleteness. Incompleteness occurs on the level of missing facts in a stream but might also include missing domain knowledge. In rule-based languages, it is handled by non-monotonic and default reasoning approaches that work under the *Closed World Assumption (CWA)* [188], thus stating that a lack of knowledge evidence that a statement is true entails its falsity. Tightly connected to CWA is weak negation, also called *negation as failure (NAF)* in logic programs, which allow the use of NAF literals, i.e., literals of the form *not a*, to express that the atomic formula *a* is not derivable by the program. The already mentioned approaches of the Datalog, DatalogMTL, and ASP-families support CWA (and possible extensions such as PCWA) as well as NAF, whereas certain restrictions are imposed regarding the usage of NAF literals in programs, e.g., stratified programs. In particular, the PCWA addresses the issue of stream data that is not (yet) available at the time when a literal *not a* in a rule is evaluated. For example, Datalog rules $last(T, E) \leftarrow occurs(T, E)$, $not\ later(T, E)$ and $later(T, E) \leftarrow occurs(T1, E), T < T1$ where the first argument encodes time, informally capture the last occurrence of an event *E*; however, when the event *E* occurs at time *T*, evaluation of the first rule is blocked since $later(T, E)$ has to be evaluated, which by the second rule may be postponed indefinitely. To avoid such blocking of the evaluation, only references to past or current data is permitted. The PCWA principle may then be applied: “If $stream(T, \dots)$ is observed in the input stream, conclude $not\ stream(T1, \dots)$, provided that $T1 < T$ and $stream(T1, \dots)$ is not entailed by the fact base augmented with the stream facts having some timestamp $T0 \leq T$.” Syntactically, PCWA can be enforced using local stratification on time. For ontology-based languages, handling incompleteness needs to be addressed differently as they work under the Open World Assumption (OWA), which means that a lack of current

knowledge leaves both possibilities for a statement, being true or false, open. This indicates that missing facts for these approaches either could be settled in a pre-processing step or be asserted by the use of existential quantifiers as available in OWL2 languages. Incompleteness may be also described using quantitative methods, where e.g., the weight of a fact expresses the certainty or probability that an observation is made, whereas the latter values 1 and 0 recover complete knowledge. Other work [73, 74] considered the handling of incomplete information in state streams for runtime verification with MTL. Here, vertices in a progression graph represent formulas, with directed labelled edges between vertices indicating that a formula can be obtained from progressing another (input) formula with a state indicated in the edge's label. A probability mass – representing the ratio of progression paths having reached a formula so far – is pushed between nodes, with terminal nodes representing verdicts (i.e., \top , \perp) and their associated probabilities, allowing for the tracking of verdict probability during progression with incomplete state information.

Constantly-varying. The aspect of constantly changing streams is less of a focus in current research due to three reasons. First, the availability of new data might not trigger the re-evaluation of the conclusions as some approaches are pull-based, whereas in push-based approaches a re-evaluation is triggered. Second, a central processing feature of these approaches is incremental updates, where the variation in the number of updated terms and not the size of updated data matters. For instance, a single ground fact deletion can trigger the re-evaluation of the full knowledge base. Note that not all mentioned approaches in this section support incremental updates. Third, variability can be considered from the semantic point of view, when the meaning of categories, concepts, or relationships changes over time. This concept drift [99] requires a DSR to include monitoring and learning components that can detect and address the drift, respectively. Automatic addressing the drift might be especially complicated since it might require updating the knowledge of a DSR, e.g., add, delete, or update rules in the case of a relational DSR. For instance, in [181], the authors equip their system with a learning algorithm to learn weights of rules and thus counteract the concept drift. The authors of [63] go even further and directly address concept drifts by applying semantic embeddings, i.e., vectors capturing KB consistency, and supervised learning to detect concept drifts in ontology streams. Nevertheless, the problem of concept drift remains largely unaddressed by modern DSR.

5 Inductive Stream Reasoning

Inductive Stream Reasoning (ISR) aims to support reasoning with new knowledge that is generated bottom-up from the data itself, which then may be used to augment deductive reasoning. In particular, this includes integrating knowledge generated by Deep Neural Networks (DNNs) from sub-symbolic inputs using machine learning algorithms, e.g., object or activity classification, but also extracting rules from stream data. In this context, dealing with uncertainty is a key issue. Applications are widespread and include critical areas such as social media analytics [25], robotics, traffic surveillance [87, 76], and autonomous driving [197]. From a sub-symbolic method perspective, a data stream can be seen as an unbounded ordered sequence of data points $S : d_1, d_2, \dots, d_i, d_{i+1}, \dots$ with $i \in \mathbb{N}$. Each data point is represented by a feature vector X_i [242]. The different data points are generated over time, and the method cannot access the entire data stream simultaneously. Usually, most of the methodologies in this context focus on the data stream classification problem, where the goal is to predict the target label y_i associated with each data point d_i whenever d_i is generated. Since existing Machine Learning solutions are not intended for use in a pure streaming scenario where the learning algorithm continuously learns from an ongoing data stream, two main areas emerged: Streaming Machine Learning (SML) [38] and Continual Learning (CL) [141].

The macro-level Stream Reasoning research question can be reformulated to the following generic meso question for ISR:

Meso (Inductive Stream Reasoning): How can we continuously make up-to-date predictions over raw data formats, e.g. sensory observations, that are constantly changing and inevitably noisy?

5.1 Make Sense

By quantifying uncertainty and creating probabilistic models, ISR systems can make more nuanced decisions based on available data streams. One of the most generic formalisations so far towards probabilistic reasoning is proposed in [91]. In another significant development, neuro-symbolic approaches [181] have been formulated to combine the generalisation ability of neural networks with the structural rigour of symbolic logic. By accommodating semantic streams embedded with probabilistic [215] and temporal dimensions [72], the ISR models become highly capable of adapting to dynamic, real-world conditions.

More concretely, given a data stream S of data points, each represented by a vector X_i , a sub-symbolic method produces a data stream of insights generated by implementing a specific learning algorithm. The potential integration of sub-symbolic methods with deductive reasoning offers various architectural possibilities. One approach involves the integration of deductive reasoning with insights generated by sub-symbolic methods. In a notable example, Kirkpatrick et al. [125] leverage sub-symbolic methods across heterogeneous data streams. The varied insights derived are then unified and employed by a deductive reasoner. Belcao et al. [32] use a similar approach to propose a bridge between Big Data Analytics and Semantic Technologies. Conversely, an alternate solution follows a different path. Here, a deductive reasoner is directly applied to a data stream, leading to continuous deduction and transformation. The transformed data becomes the canvas for sub-symbolic methods to apply inductive reasoning and yield outputs, as demonstrated by Barbieri et al. [26]. To make this integration between deductive and inductive reasoning more concrete, let's introduce a practical problem.

► **Example 4 (Taxi cont'd).** Suppose a taxi driver must answer questions like “*What museum can I reach in less than 25 minutes leaving at this exact moment?*” To solve this problem, Della Valle et al. [76] use different types of information. Firstly, the work retrieves monuments, attractions, exhibitions, and events in the city of Milan from different open data in RDF format. It also adds the topology of the city's streets, detections of traffic sensors and weather information. For each traffic sensor, a specific sub-symbolic method is applied. Particularly, the authors train Recurrent Neural Networks to forecast traffic. The different sensors' predictions are propagated to generalise beyond the sensors' locations by exploiting the street graph topology. Ultimately, a deductive stream reasoner comes into play, addressing user queries through deductive reasoning. This reasoner seamlessly integrates RDF data and traffic predictions obtained from sub-symbolic methods.

5.2 Taming Volume

The massive volume of streaming data in real-time from various sources is another issue that ISR aims to tackle. A robust sub-symbolic streaming method should be easily embeddable in a stream processing pipeline capable of running multiple concurrent queries on big data volumes while dealing with high update loads. In terms of data volume, it could range from megabytes in

nanoseconds to terabytes in minutes, depending on the specific requirements of an application. For example, autonomous cars generate around 25 Gigabytes of data per hour including 4-6 radars (0.1-15Mbit/s), 1-5 LIDARs(20-100/Mbit/s), 6-12 cameras (500-3500Mbit/s) and under 0.01Mbit/s sensors such as Ultrasonic, Vehicle motion, GNSA and IMU. Moreover, most of the systems have to deal with multiple concurrent queries on big data volume paired with high update loads generated continually from multiple streams from multiple sources. On top of that, it depends on the reactivity constraint. It can be MB in nanoseconds, GB in seconds, TB in minutes. As mentioned above, it should be impossible to tame that volume, ignoring the streaming nature of the data.

5.3 Taming Variety

Streaming sub-symbolic methods can support different natures of data. SML is usually applied to structured data streams containing data points with tens of features. Classical real-world benchmarks include Airline [120], containing flight arrival and departure details; Forest Cover Type [42], representing forest cover types of specific geographical areas based on different attributes determined by the US Forest Service; and KDDCup99 [210], including data for intrusion detection in a network. On the contrary, CL usually deals with unstructured data like images. Each data point can contain hundreds of thousands of features. Standard benchmarks include streaming versions of the most known computer vision benchmarks (MNIST [237, 108] or CIFAR [147]). An interesting case is represented by the OpenLORIS [201] benchmark, which provides a comprehensive set of visual, inertial, and odometry data captured with real robots in authentic scenes. The goal of the learning model can be scene understanding or evaluating Simultaneous Localisation and Mapping.

Moreover, more complex multi-model data streams can be supported, such as in autonomous driving applications, where data originates from various sensors, each providing a unique lens through which to view the environment. For instance, as shown in [197], an autonomous vehicle may use radar, lidar, and cameras to understand its surroundings, requiring the reasoning system to integrate and make sense of this disparate data using deep neural networks and other signal processing components to lift these raw data into symbolic forms to symbolic solvers or reasoners. In essence, DSRs above can then be used as an underlying component to make logical decisions regarding the observed data.

5.4 Taming Velocity

One of the most critical aspects of ISR is its near real-time decision-making capabilities. For instance, in traffic surveillance applications, the system must make immediate decisions based on incoming data. It cannot afford to wait for the entire data set before beginning the analysis. This is especially true for trajectory predictions, where potential paths or actions are inferred even before complete data is received. Additionally, as soon as data becomes available, immediate insights are formulated and communicated, ensuring minimal delay. However, this challenges existing machine learning and reasoning systems, which may not have been designed to handle the high-speed, ever-changing nature of data streams.

Regarding the sub-symbolic methods, an SML learning model aims to predict the label \hat{y}_i whenever a new data point is generated. It assumes that the actual label y_i arrives after casting the prediction. The model is updated incrementally whenever a new y_i is available. SML models can use each data point only once. In Batch Incremental Learning (BIL), data points are accumulated in fixed-size batches containing tens of them [186]. The model is updated once the mini-batch fills up. SML prioritizes computational efficiency in time and memory. Following BIL's direction,

CL assumes the data points will be grouped into large batches called experiences. However, each experience e_i can contain thousands of data points (rather than tens), randomly accessible as many times as the model requires. A CL strategy can process each e_i for as long as necessary before accepting a new experience.

Concept drift. Concept drift is a critical aspect associated with the evolving environment of data streams. The traditional Machine Learning assumption that data is independent and identically distributed does not hold in this context, where data can change its distribution. A concept is the unobservable random process producing data points [97]. Concept drift is a phenomenon in which the statistical properties of a domain change over time in an arbitrary way [148]. We can categorise concept drift into two primary types: virtual and real. Virtual concept drifts occur when the probabilities $P(X|y)$ or $P(y)$ change. Real concept drifts happen, instead, when there is a change in the $P(y|X)$ probability. Therefore, a virtual concept drift does not affect the class boundary, while a real concept drift introduces a change. Additionally, in cases of abrupt drifts, the new concept instantaneously replaces the old one. Conversely, the new concept gradually or incrementally replaces the old in gradual and incremental drifts. Lastly, it's also possible for concepts to re-occur over time. SML assumes the distribution within a concept to be fixed. SML literature puts a strong effort into automatically detecting concept drifts. These solutions can deal with all concept drifts. Conversely, CL assumes that each new experience introduces an abrupt concept drift. For this reason, it does not use concept drift detectors.

Temporal dependence. Numerous data streams exhibit dependencies on their past values [40, 242]. For instance, an attribute value may result from an auto-regressive transformation applied to preceding instances, such as the fluctuation in commodity prices like electricity [36] or the evolution of weather conditions [84]. Modelling the evolving temporal patterns, e.g., trends or seasonalities, can be challenging, and traditional methods assuming independence between observations are unsuitable [241]. Consequently, addressing this intricate issue requires the development of specialised methods capable of capturing the dependencies inherent in the data. While there has been significant emphasis on detecting concept drifts and developing techniques to adapt to such changes, the issue of independence has received comparatively less attention. Approaching this matter from a SML standpoint, the filtering task within a sequential-state space model, such as Kalman Filters, emerges as a promising avenue for managing concept drift and temporal dependence [240]. Similarly, applying CL methods on Recurrent Neural Networks to learn sequences within a data stream presents another encouraging approach to address this complex problem, as evidenced by the introduction of Continuous Progressive Neural Networks [103].

The challenge of temporal evolution extends to the integration between ISR and DSR. Envisioning an inductive reasoner, such as SML or CL models, processing data and recognizing the entities for input to a deductive reasoner introduces potential issues when entities undergo evolution over time, necessitating adaptive responses from inductive reasoners. In such instances, the challenge may not be adapting to a concept drift but learning the entity's natural evolution. Consequently, accounting for temporal coherence within the data streams becomes crucial. Notably, intriguing benchmarking datasets are showcasing temporal dependencies within the Continual Learning (CL) field. Recent studies [143, 235] introducing datasets with inherent temporal aspects and concepts like temporal distribution shift and coherence, underscore a growing community interest in this direction.

Learning goals and evaluation. Despite both SML and CL learning from data streams and managing concept drifts, they have different objectives. The main goal of SML is to detect concept drifts automatically and quickly adapt to new concepts. An SML model must learn fast, react

quickly to concept drift, and perform well on the current concept. It is also subjected to strict constraints on time and memory consumption. Conversely, CL addresses the stability-plasticity dilemma [151]. When learning new experiences, the model may forget what it has learned during the previous ones. The ability to remember past knowledge is called stability, while learning new knowledge is called plasticity. Too much stability could lead to difficulties in learning new knowledge. Conversely, too much plasticity may lead to forgetting past knowledge and raising the problem known as catastrophic forgetting [132]. The goal is to achieve a trade-off between stability and plasticity. A CL strategy must perform well on all the seen experiences from the first to the current one. This difference in objectives is reflected in different evaluation procedures. SML does not distinguish between training and test data, and each data point is used for both purposes. The evaluation protocols usually include a prequential evaluation [98]. Each time a new data point is generated, the SML model predicts its label \hat{y}_i . When the actual label y_i is available, the protocol updates the evaluation metric (usually accuracy or Cohen's Kappa Score) and then trains the model on d_i . Conversely, CL evaluates the model's ability to mitigate forgetting and learning new experiences [141]. Each experience e_i is split into a training set D_i^{train} and a test set D_i^{test} . All the D_i^{test} are always available, while the D_i^{train} are provided over time. Different types of metrics exist [141]. Accuracy is usually evaluated, after each experience's training, how the model performs on the current experience's test set and all the previous experiences' test sets. Average accuracy evaluates the model's overall performance after the last experience's training by considering the average accuracy on all the D_i^{test} . The Backward Transfer Metric measures the stability of the model and, more in general, how the final version of the model has improved or decreased the performance of the previous versions. After training on the last experience, for each previous experience e_i , it subtracts the accuracy of the model trained on the experience e_i from the one of the current model tested on D_i^{test} . A negative value indicates forgetting. Finally, the Forward Transfer Metric measures how the training on the current experience is useful also for learning the next experience. For each experience e_i , it subtracts the accuracy of a random model tested on D_i^{test} from the one achieved by the model after the training on e_{i-1} . A positive value indicates that the current training positively affects the performance for the next experience.

5.5 Domain Complexity

The main focus of inductive reasoning is on the data and the signals it carries. When extracted, such signals can be used to construct the domain. In this context, the domain complexity is usually a *tuned* parameter: it is up to the scientist or engineer to determine the adequate complexity of the model that will fit the data. Existing solutions try to build models of various complexities. Lecue and Pan [138] propose an approach that extracts association rules from streaming data. Balduini et al. [19] study how to extend inductive reasoning methods to a streaming scenario. The resulting system uses RDF streams to create matrices, which are fed to an inductive reasoner, SUNS, to recommend items.

Learning algorithms. For the sub-symbolic part, different choices are possible. SML usually applies simple models, often based on Statistical Machine Learning. Frequency-based methods track feature frequencies and calculate posterior probabilities using Bayes's theorem. Neighbourhood-based techniques identify neighbours for new samples based on distance, often using a sliding window to manage recent instances. Tree-based classification algorithms are streaming versions of decision trees that use the Hoeffding bound [117] for incremental split node decisions. Techniques like Hoeffding Adaptive Trees (HAT) [37] address concept drift, incorporating concept drift detectors. Ensemble-based methods combine predictions from individual models to enhance generalisation with well-known techniques like Online Bagging, Leveraging Bagging, and Adaptive

Random Forests [107]. Conversely, CL usually applies more complex models based on Deep Learning. It employs three main categories of strategies [141]. Replay approaches (e.g., [176, 183, 5]) store a subset of examples encountered during training in external memory to combat forgetting. They blend this memory with the current data during each iteration to update the model. In this context, Generative Replay methods (e.g., [202]) employ generative models to recreate past examples as needed, eliminating the need for external memory. Regularisation strategies (e.g., [142, 127]) bolster the loss function with additional terms to enhance model stability and mitigate forgetting. They can, for instance, restrict changes in parameters crucial for previous data or enforce consistent network activation over time. Architectural strategies (e.g., [145, 193, 200]) adapt the model's architecture to incorporate new knowledge while minimising forgetting. Popular techniques include expanding the number of layers or units over time and compressing or freezing previous model components. Hybrid approaches (e.g., [147, 187, 198]) that combine elements from more strategy families are often highly effective.

Frameworks. Various frameworks facilitate the application of SML and CL algorithms. Notably, almost all existing frameworks are mainly used for research, as they still have significant limitations for industrial applications. The Massive Online Analysis (MOA) framework [39] presents a broad spectrum of algorithms designed for multiple tasks related to data stream analysis. MOA's tasks encompass classification, regression, multi-label, multi-target, clustering, outlier detection, concept drift detection, active learning, and more. In addition to learning algorithms, MOA offers data generators (e.g., AGRAWAL, Random Tree Generator, and SEA), evaluation methods (e.g., periodic holdout, test-then train, prequential), and statistics (CPU time, RAM-hours, Kappa). The Scalable Advanced Massive Online Analysis (SAMOA)[155] is both a framework and a library that combines stream mining and distributed computing (i.e., MapReduce). SAMOA allows users to abstract the underlying stream processing execution engine and concentrate on the learning problem. It provides adapted versions of stream learners for distributed processing, including the Vertical Hoeffding Tree algorithm[129], bagging, and boosting. Vowpal Wabbit (VW) is an open-source machine learning library featuring an efficient and scalable implementation with several learning algorithms. VW has demonstrated its capability by learning from a tera feature dataset using 1000 nodes in approximately an hour [3]. StreamDM, an open-source framework for big data stream mining, utilizes the Spark Streaming extension of the core Spark API. One notable advantage of StreamDM over existing frameworks is its direct integration with the Spark Streaming API, which efficiently handles complex issues arising from the underlying data sources, such as out-of-order data and recovery from failures. There has been a significant recent development in SML algorithms for Python, with the River package [154] being particularly noteworthy. River supports various ML tasks, including regression, classification, and clustering. Moreover, River is versatile enough for ad hoc tasks, such as computing online metrics and detecting concept drift. Finally, Avalanche [146] deserves mention as the first experiment of an end-to-end Library for reproducible CL research and development. It encompasses implementing state-of-the-art CL strategies, standard benchmarks, evaluation metrics, and evaluation scenarios.

5.6 Data Quality

ISR has the potential, to address data imperfections, such as incompleteness and noise, that can have a disruptive effect on Deductive Stream Reasoning. Data incompleteness arises in sensor networks due to factors like sensor battery depletion or network link interruptions. In social media, instead, it arises due to limited sampling rates in social stream APIs or – in a certain sense luckily – because conversations also occur outside social networks. Noise issues encompass sensor network imperfections or operational deviations. In processing unstructured data such as text, sounds,

images and videos, it occurs due to low accuracy in tools used for analysing them such as the inability to catch irony, difficulties in transcribing phonetically ambiguous words, or in evaluating occlusions in object detection and tracking.

DSMS and CEP have traditionally handled noise [68]. Two main types of noise have been identified, affecting content and temporal annotations. Content noise pertains to inaccuracies in data from sensors or human interactions, potentially leading to incorrect conclusions. Statistical methods can manage noise in simple schemas, but more complex schemas require advanced techniques such as [137]. Researchers can explore streaming machine learning approaches to process noisy data, coupled with deductive reasoning techniques like inconsistency repair [10, 51], and belief revision [192, 190]. Temporal annotation noise involves out-of-order data items, particularly when multiple streams with different time annotations are involved. Solutions exist for handling temporal noise, with room for semantic enhancements [203, 144]. Addressing temporal noise may involve aligning diverse temporal annotations from different sources. Existing solutions can be adapted, with semantic enhancements offering promising opportunities [51].

In summary, ISR must increase efforts in tackling imperfections like incompleteness and noise, necessitating innovative solutions and approaches for both content and temporal issues.

6 Discussion

This paper *grounds* the Stream Reasoning research, by providing a clear overview of its different constituent research areas, and explaining how each of these areas target its different dimensions. For over a decade, practitioners from different research communities have contributed to Stream Reasoning research, each from within their perspective and background. Since 2015, researchers in these different communities have organised the *Stream Reasoning Workshops*, a recurring event with the purpose of sharing perspectives, challenges, and experiences around Stream Reasoning topics.

This paper provides an overview of the main research contributions discussed during the *Stream Reasoning Workshops*. Moreover, this paper provides a crystallisation of how each of the different research areas within Stream Reasoning perceive and tackle the Stream Reasoning research dimensions. By understanding how the different areas differ and relate and how they perceive the various Stream Reasoning research dimensions, this paper *grounds* the Stream Reasoning research. We conclude with a discussion of the take-away messages and open challenges for the next years.

6.1 Discussion of the Research Dimensions

We will now discuss how the different areas differ or relate in tackling the dimensions introduced by the original SR research questions. Table 3 summarises the discussion by providing an overview of how the different areas target the research dimensions.

- **Making Sense:** Even though each area has a different focus when *making sense* of the data streams, e.g., Stream Processing focuses on Continuous Querying, RSP/SLD on Continuous Data Integration and Querying, DSR on incremental materialisation, model checking and planning, each area has a *continuous* component when making sense and is typically done through some kind of query. In Stream Processing this is an SQL-like language, in RSP/SLD a dialect of SPARQL, while in DSR this is mostly done through rules.
- **Taming Volume:** Volume has not been the main point of focus for any of the approaches, except for Stream Processing which incorporates techniques to scale horizontally. The mechanisms to tame variety, as in RSP/SDL, or to incorporate rich domain complexity, as in DSR, have a negative impact on the volume of data that can be processed.

■ **Table 3** Continuous Querying (CQ); Consistency (C); Data Integration (DI); Model Checking (MC); Materialisation (MAT); Planning (P); Clustering (CLU); Classification (CLA); Temporal Logic (TL); Incompleteness (I); Noise (N); Volatility (V).

Dimension	SP	SLD	DSR	ISR
Making Sense	CQ	C, DI	MAT/MC/P	CLU/CLA
Velocity	Sub-millisecond	Milliseconds	Seconds	Milliseconds
Variety	Relational, Document	RDF	Relational	Multimodal
Volume (Scale Up/Out)	Yes/Yes	Yes/Limited	Yes/No	Yes/Edge
Domain Complexity	Data Schema	RDFS+	OWL2, ASP, TL	Variable
Data Quality	I, N	I	I, C	I, V, N

- **Taming Variety:** Stream Processing requires a manual mapping to the used relational schema, which is not a flexible approach and is typically not well-suited for data integration purposes. RSP/SLD are able to solve a data integration problem in a continuous fashion by relying on RDF and the extension of the Semantic Web stack and is thus the best-suited approach for handling variety. DSR and ISR typically map to RDF to increase the support of data variety, but do not directly build upon the Semantic Web stack. ISR tames variety in a different way by supporting multi-modal streams, e.g. integrating video with numerical sensor readings.
- **Taming Velocity:** All approaches use some form of *windowing* to deal with data streams, however, the requirements in terms of responsiveness differ, Stream Processing focuses on sub-milliseconds latency, RSP/SLD and ISR focus on milliseconds latency, while DSR is satisfied with latency in terms of seconds. Stream Processing is the fastest, as it does not require any overhead to perform data integration such as RSP/SLD, checking models and incorporating complex domains as in DSR, or performing predictions as in ISR.
- **Domain Complexity:** DSR allows the incorporation of the most complex domain knowledge, at the cost of performance. RSP/SLD supports little domain complexity in order to prioritise responsiveness. Although the focus is growing, Stream Processing has limited support for domain complexity, rather than focusing on volume and velocity.
- **Data Quality** has not been properly addressed by the different approaches. ISR has valuable solutions for *veracity* through predictions, while DSR can handle *incompleteness* very well by inferring missing facts in a deductive manner through the incorporation of domain knowledge. Stream Processing solutions to deal with *constantly varying* data, which have been adopted to some extent by SLD and RSP. Different areas have focused to some extent on the different aspects of data quality, however, none of the approaches has targeted them simultaneously.

It is clear that each area has tackled different aspects of the original SR research dimensions. Some dimensions have been tackled by all of them, e.g. Velocity, while others have received more attention in one of the areas, e.g. Variety in RSP/SLD, Domain Complexity in DSR, and incompleteness in ISR. In order to realise the SR vision, cross-pollination between different areas is needed to cover all the dimensions simultaneously.

6.2 Overlapping Approaches

Even though the large body has been done in the distinct areas of the SR research, there has been research conducted that combines ideas from multiple areas:

Streaming Linked Data & Deductive Stream Reasoning

From within RSP/SLD, there has been a quest to increase the expressiveness of the reasoning capabilities and thus increase the domain complexity. This has led to an investigation into how more expressive reasoners can be combined or integrated with RSP engines.

Morph-Streams [56] and OntopStream [32], which employ an OBDA approach can support OWL2 QL ontologies through their rewriting regimes. RoXi [44] is a recent effort to increase the domain complexity of RSP engines to OWL2 RL, through various optimisations to enable Datalog reasoning over RDF streams in an efficient fashion, e.g., through efficient maintenance of the materialisation in the window [79] or pruning of the reasoning rules [47].

However, in general, there is a mismatch between the complexity of more expressive reasoning algorithms and the change frequency of the data streams that RSP engines try to tackle. To solve this mismatch, the vision of Cascading Reasoning [204] emerged, which suggests a layered approach of processing and reasoning engines where the lower layers process the high-velocity data with techniques of limited complexity, going up in the layers, the amount of data decreases while the complexity of data increasing, ultimately resulting in support for expressive reasoning over high-velocity streams. There have been first realisations in this area, such as StreamRule [152], which combines the CQELS engine for RSP with ASP reasoning, and Streaming Massif [48], which combines C-SPARQL with the HermiT reasoner for Description Logic and features complex event processing capabilities. However, these initial approaches still require manually defining the processing at each layer and thus do not constitute a generic approach to define the reasoning and processing across various layers. Some interesting early developments in that area aim to rewrite registered continuous queries and to prune datalog rules in order to push the processing to lower layers in the hierarchy [45].

Streaming Linked Data & Inductive Stream Reasoning

In the early days of Stream Reasoning, there were several attempts to combine Inductive and RSP for the analysis of social media streams. The first seminal work [25] introduced a pipeline combining the deduction of a C-SPARQL engine [24] with a long and with a short window, whose contents is transformed into a matrix factorisation system, in order to recommend links in a bipartite graph that pairs users to movies. The combination of a long and short window allows to capture both long lasting knowledge and hype effects. A similar approach was further developed in [19]. Follow-up work led to demonstration of the applicability of this schema to venue recommendation [18].

CQELS 2.0 [136] extends the CQELS engine with more powerful inductive capabilities such as Deep Neural Networks, and it allows for the fusion of various multi-modal data streams. For example, by fusing object detection on video streams with sensor readings of location and velocity and by converting the results to the RDF model, the streams can be queried continuously in order to solve multi-object tracking in a declarative fashion.

Interestingly, the approaches in this overlap augment the data itself and extend the query language in order to support query answering over certain predictions as a result of the inductive part.

Deductive Stream Reasoning & Inductive Stream Reasoning

Above we presented ISR as a method to generate new knowledge that can be used in combination with deductive reasoning. A natural question is then a fruitful combination of symbolic deductive reasoning techniques with inductive techniques. Induction may further be interrelated with abductive reasoning for learning, by generating hypothetical facts from which new complex knowledge may be inferred, possibly in a cycle [123]. For such learning, dealing with uncertainty is an important aspect. Specifically, in the PrASP [161] approach programs may inductively learn from data streams and can be incrementally evaluated. For Stream Reasoning programs in [181], merely the weights of rules as independent probabilities can be learned; that work showed how the potential of combining DSR and ISR in the realm of object tracking under uncertainty in traffic monitoring, demonstrated in [205], can be pushed to real-time performance with proper Stream Reasoning infrastructure. These hybrid approaches are also particularly important in applications like robotics, where generalisation and structured knowledge are vital [214].

6.3 Open Challenges

In terms of open challenges, we discuss the open opportunities for each research area that should be solved in the next years in order to push the field closer to the true realisation of the Stream Reasoning vision.

6.3.1 Stream Processing

With the availability of several large-scale technological infrastructures for stream processing, which have been successfully deployed in multiple industries, it may be tempting to consider stream processing as a solved problem. We warn the reader to make such a conclusion as we believe that there are still several important challenges that need to be addressed.

First of all, current solutions, e.g., **Apache Flink**, are designed to be general-purpose solutions. Such solutions sacrifice some performance to support a wider range of applications. Since approaches for RSP, SLD, DSR, and ISR can be computationally demanding, we argue that an important challenge is:

- *How can we optimise current general-purpose solutions for stream processing to support more efficient reasoning applications?*

An alternative approach consists of improving current reasoning solutions exploiting what has been learned while developing the current state-of-the-art for generic stream processing. Hence another important challenge is:

- *How can we improve the state-of-the-art for stream reasoning adopting the best practices in large-scale stream processing?*

In both cases, the challenges call for a deeper collaboration between members of various communities. This collaboration has a huge potential, not only to solve the problem at hand but also to discover new research avenues that can benefit both sides.

6.3.2 Streaming Linked Data

There are still various open challenges in the realm of RSP and SLD. We summarise the most important ones in the form of micro questions:

- *How can we increase the expressivity of the ontologies when supporting reasoning in SLD?* Most approaches provided limited to no reasoning capabilities or simple regimes such as RDFS. The reason is that there is a mismatch between the complexity of the algorithms to perform more expressive reasoning and the responsiveness and low latency requirements of many of the

use cases targeted in RSP. One vision to mitigate this is the idea of Cascading Reasoning [204], which suggests a layered approach of processing and reasoning engines where the lower layers process the high-velocity data with techniques with limited complexity, going up in the layers, the amount of data decreases and the complexity of data increasing, ultimately resulting in supporting expressive reasoning over high-velocity streams. There have been first realisations in this area, such as Streaming Massif [48], however, it is still required to manually define the processing at each layer.

- *How can virtual processing of RDF Streams be integrated with techniques that process “real” RDF Streams?* Both techniques have their benefits, but they have not been combined in order to reap the benefits of both approaches. Doing so would allow various optimisations as the conversion from virtual to “real” RDF Streams could be done in an optimal fashion, while integrating different sources of data. This could lead to more flexible RSP engines, able to adapt to RDF and non-RDF streams through virtualisation. The question of seamless mapping of these sources to WoT or IoT ontologies can also be beneficial, especially for environments where native RDF is not necessarily the most efficient option.
- *How can we use SLD as the basis for autonomous computing on the Web in rapidly changing environments?* The *linked* nature of SLD has only been exploited to a limited extent [54]. However, the potential for autonomous agency in stream processors [218] is high in terms of distribution of query processing load, and local processing capabilities – e.g., for WoT environments. For this to be properly implemented, it would be necessary to specify agent-based primitives such as goals, intentions, or beliefs, related to the capabilities of RSP engines. Moreover, it would be necessary to include scheduling and coordination mechanisms for enabling efficient interactions among the autonomous RDF stream engines [209]. Moreover, when data streams are managed using the web as the processing platform, new problems may emerge, in particular when publishing personal data which may contain sensitive information. It is therefore important to account for the privacy issues that may arise [77].
- *How can agents on the web collaborate to answer continuous queries?* Although the vision of interconnected RSP engines has been discussed in the past [80], its realisation is yet to become a reality. It will be necessary to further investigate decentralised query processing for stream environments, as well as explore how to formalise the semantics of cascading stream processing, including how temporal aspects are affected by distributed query answering.
- *How can we continuously query RDF streams that have a much higher change frequency than the milliseconds change frequency that SLD solution can currently handle?* To cope with streaming loads under high demand as it is currently done in non-RDF systems, hybrid approaches that combine RSP and native stream data processing can lead to promising solutions. This would require further exploring optimisation opportunities, as well as exploiting OBDA-based approaches using underlying high-performance native engines.
- *How can we perform data integration continuously over data streams of different formats?* Although this topic has been addressed through virtualisation and materialisation to some extent, most streams on the Web are not RDF nor follow RDF-like formats. Moreover, in many cases there is no interest in necessarily transforming all of the contents into RDF. Hence, there is a challenge to manage hybrid RDF streams, which could be processed by engines that can handle different stream models.

6.3.3 Deductive Stream Reasoning

There are numerous challenges related to the development of DSRs. Among the most important ones are undoubtedly those concerning the efficiency of the reasoning process so that it can meet the requirements of a typical streaming scenario. In this context, we can distinguish two main challenges:

- *How can we shorten the response time of query answering with current formalisms such as LARS and DatalogMTL?* Examples of works in this category are the ones that focus on maintaining the materialisation after updates [27], or the ones that combine materialisation with other techniques [230].
- *Can we find a good balance between the expressivity of the reasoning while still maintaining a high throughput?* For instance, works in this category are the ones that restrict existing formalisms to allow a faster execution [27] or the ones that provide extensions to support higher expressivity without compromising performance [223]. Another type of approaches falling into this category is focused on formal analysis of the input stream for data and/or reasoning parallelisation [179, 92].

Another important challenge consists of improving the usability of DSRs. Currently, the usage of a DSR requires a good understanding of the underlying technology *and* a proper way to represent the domain using a formal language. It may be challenging to meet these requirements in practice. Hence, two important research directions are:

- *How can we automatically capture knowledge in a formal language so that DSRs can use it to reason over the data streams?* Although inductive logic programming systems are available for popular languages like Prolog or ASP, as far as we know, this research question has not been properly investigated yet.
- *How can we communicate to users that are not familiar with formal languages why a reasoner has returned some particular information?* This topic has become particularly important since AI is being adopted in an increasing number of domains. Like the previous question, this topic of research on streaming scenarios has not yet been studied, although in this case there are numerous works in static contexts that can be used as a starting point. Furthermore, recent developments in large language models open an interesting perspective for user-friendly communication between formal-language based systems and humans with little formal background and training. Current attempts to exploit such models in fields like planning, argumentation, and temporal logic may be lifted to the DSR setting.

Finally, another important topic of research consists of combining the power of logic-based reasoning with techniques that can deal with uncertain data. This combination will allow us to counter more effectively all the challenges related to data quality. Moreover, uncertainty is inherently present in many domains and dealing with it is becoming increasingly important as more and more machine learning techniques are being introduced in key decision processes.

Also in this case we identify two main research directions:

- *How can we include uncertainty, either aleatoric or epistemic, into the query answering process?* This question can be investigated by trying to adopt existing approaches to uncertainty such as the possible world semantics [207] into a DSR or by the development of completely new frameworks tailored to the needs of streaming scenarios. The PrASP system [161] discussed in Section 4.5 represents a preliminary attempt to rely on possible world semantics to handle uncertainty in an ASP-based stream reasoner, but the limited scalability makes it unsuitable for real-world scenarios. This calls for the investigation of better sampling strategies and uncertainty reasoning that does not necessarily rely on external solvers, but specifically tailored to work with data streams.
- *How can we exploit stream reasoning with uncertainty to formulate what-if scenarios?* This problem is particularly important when a DSR is used to help decision processes when errors can be very costly. Reasoning to determine not only what is true now, but also what can/cannot be true in the near future can be invaluable as it could help to prevent malfunctioning in power plants or disasters in crowd management. Unfortunately, as far as we know this topic has not been investigated yet in a streaming setting.

The aforementioned list is by no means a complete enumeration of all issues. There are also several other problems which are equally valuable and deserve much attention by the community. For instance, currently we lack solid and fair evaluation frameworks to compare the performance of DSRs. In other more mature fields, the community has agreed on establishing an independent team, which include many representatives from industry and academia, to define comprehensive benchmarks. Examples of such initiatives are the TPC-H benchmark suite [220] and the LDBC consortium [208]. Doing so also for DSR systems or, more in general, for stream reasoning in general is a natural next step. Another important problem relates to the development of tools that are beyond proof-of-concepts and which are robust enough to be used outside the community.

6.3.4 Inductive Stream Reasoning

By addressing challenges related to real-time decision-making, data variety, veracity, and massive volumes, an ISR system aims to offer a robust framework for reasoning in dynamic environments. Meso-level questions, like how to ground multi-modal data into high-level reasoning, still require further exploration and implementation to be used in killer applications such as autonomous vehicles and robotics. Likewise, the application of model checking under conditions of uncertainty and incompleteness remains a vital area for future research. Therefore, as we move towards an increasingly connected and data-intensive world, the role of inductive stream reasoning as a harmonising factor between traditional models and complex reality is set to become even more crucial. In this context, we identify the following research challenges:

- *How to ground multimodal sub-symbolic data streams into high-level reasoning?* The neural-symbolic field proposes several solutions to connect sub-symbolic data and high-level reasoning facts and rules, such as DeepProblog [149] and NeurASP [234]. However, there are only a few works, such as [206] and [181] considering streaming aspects of data. Most of them only address this problem in very narrow domains or specific types of data and rules. Hence, this calls for a systematic investigation of stream-first approaches, including learning to inference phases of grounding multimodal stream data into high-level stream reasoning.
- *How to implement performant and scalable ISR systems for real world applications?* Most of the motivated applications for ISR have very critical requirements for low-latency in conjunction with data volume like pointed out in Section 5. However, so far, there is no implementation that can deal with the data scale of the targeted applications, e.g., autonomous driving, robotics, and automation. To make ISR usable for such targeted applications, there is an urgent need for more systems taking operational requirements seriously to make real-life impact to relevant industries.
- *Can we perform model checking under uncertainty applied to signals modelled as streams representing the physical world, when that incomplete and rapidly-available information is assumed to evolve over time as the result of external processes?* Model checking has the potential to introduce formal verification in stream processing pipelines [31], thus contributing to verifying correctness of processing results, identifying problematic stream processes, or providing resource allocation information. Run time verification of observations and events in data streams can also benefit from these works [109], although current approaches do not fully incorporate uncertainty and domain knowledge as it is the case in ISR.

6.4 Summary

For more than a decade, researchers and practitioners from different communities have contributed to advance Stream Reasoning, each from within their area and perspective. Since 2015, members of these communities have organised the *Stream Reasoning Workshop*, an annually recurring event

with the purpose of sharing perspectives, challenges, and experiences around the Stream Reasoning topic. This paper not only reviews the main research contributions discussed at these workshops, but provides a clear overview of the different areas that constitute the research field. Furthermore, it crystallises how each of these areas perceives and tackles the various dimensions of Stream Reasoning research. By understanding the views of these areas and how they relate and differ, the Stream Reasoning research is *grounded* in sense. Future research efforts in the areas may be aligned and benefit from each other based on our analysis, leading to powerful stream reasoning technology and systems that are urgently needed in an ever streaming world.

References

- 1 Zainab Abbas, Vasiliki Kalavri, Paris Carbone, and Vladimir Vlassov. Streaming graph partitioning: An experimental study. *Proc. VLDB Endow.*, 11(11):1590–1603, 2018. doi:10.14778/3236187.3236208.
- 2 Lorenzo Affetti, Alessandro Margara, and Gianpaolo Cugola. Tspoon: Transactions on a stream processor. *J. Parallel Distributed Comput.*, 140:65–79, 2020. doi:10.1016/j.jpdc.2020.03.003.
- 3 Alekh Agarwal, Olivier Chapelle, Miroslav Dudík, and John Langford. A reliable effective terascale linear learning system. *J. Mach. Learn. Res.*, 15(1):1111–1133, 2014. doi:10.5555/2627435.2638571.
- 4 Muhammad Intizar Ali, Feng Gao, and Alessandra Mileo. Citybench: A configurable benchmark to evaluate rsp engines using smart city datasets. In *International semantic web conference*, pages 374–389. Springer, 2015. doi:10.1007/978-3-319-25010-6_25.
- 5 Rahaf Aljundi, Eugene Belilovsky, Tinne Tuytelaars, Laurent Charlin, Massimo Caccia, Min Lin, and Lucas Page-Caccia. Online Continual Learning with Maximal Interfered Retrieval. In *NeurIPS*, pages 11849–11860, 2019. URL: <https://proceedings.neurips.cc/paper/2019/hash/15825aee15eb335cc13f9b559f166ee8-Abstract.html>.
- 6 Denise Anglica, Giovambattista Ianni, Francesco Pacenza, and Jessica Zangari. Integrating aspbased incremental reasoning in the videogame development workflow (application paper). In Michael Hanus and Daniela Incezan, editors, *Practical Aspects of Declarative Languages - 25th International Symposium, PADL 2023, Boston, MA, USA, January 16-17, 2023, Proceedings*, volume 13880 of *Lecture Notes in Computer Science*, pages 96–106. Springer, 2023. doi:10.1007/978-3-031-24841-2_7.
- 7 Darko Anicic, Jean-Paul Calbimonte, Óscar Corcho, Daniele Dell’Aglío, Emanuele Della Valle, Shen Gao, Alasdair J.G. Gray, Danh Le Phuoc, Robin Keskisärkkä, Alejandro Llaves, Alessandra Mileo, Bernhard Ortner, Adrian Paschke, Monika Solanki, Roland Stühmer, Kia Teymourian, and Peter Wetz. RDF Stream Processing: Requirements and Design Principles. Technical report, W3C RSP Community Group, 2015. URL: https://streamreasoning.org/RSP-QL/RSP_Requirements_Design_Document/.
- 8 Darko Anicic, Sebastian Rudolph, Paul Fodor, and Nenad Stojanovic. Stream reasoning and complex event processing in ETALIS. *Semantic Web*, 3(4):397–407, 2012. doi:10.3233/SW-2011-0053.
- 9 Arvind Arasu, Shivnath Babu, and Jennifer Widom. The CQL continuous query language: semantic foundations and query execution. *VLDB J.*, 15(2):121–142, 2006. doi:10.1007/s00778-004-0147-z.
- 10 Marcelo Arenas, Leopoldo E. Bertossi, and Jan Chomicki. Consistent query answers in inconsistent databases. In Victor Vianu and Christos H. Papadimitriou, editors, *Proceedings of the Eighteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, May 31 - June 2, 1999, Philadelphia, Pennsylvania, USA*, pages 68–79. ACM Press, 1999. doi:10.1145/303976.303983.
- 11 Michael Armbrust, Tathagata Das, Joseph Torres, Burak Yavuz, Shixiong Zhu, Reynold Xin, Ali Ghodsi, Ion Stoica, and Matei Zaharia. Structured streaming: A declarative api for real-time applications in apache spark. In *SIGMOD*, 2018. doi:10.1145/3183713.3190664.
- 12 Alessandro Artale and Enrico Franconi. Temporal description logics. In *Handbook of Temporal Reasoning in AI*, pages 375–388. Elsevier, 2005. doi:10.1016/S1574-6526(05)80014-8.
- 13 Alessandro Artale, Roman Kontchakov, Alisa Kovtunova, Vladislav Ryzhikov, Frank Wolter, and Michael Zakharyashev. First-order rewritability of ontology-mediated queries in linear temporal logic. *CoRR*, abs/2004.07221, 2020. doi:10.48550/arXiv.2004.07221.
- 14 Alexander Artikis, Marek J. Sergot, and Georgios Paliouras. An event calculus for event recognition. *IEEE Trans. Knowl. Data Eng.*, 27(4):895–908, 2015. doi:10.1109/TKDE.2014.2356476.
- 15 Ahmed Awad, Riccardo Tommasini, Samuele Langhi, Mahmoud Kamel, Emanuele Della Valle, and Sherif Sakr. D²ia: User-defined interval analytics on distributed streams. *Inf. Syst.*, 104:101679, 2022. doi:10.1016/j.is.2020.101679.
- 16 Shivnath Babu and Jennifer Widom. Continuous queries over data streams. *SIGMOD Rec.*, 30(3):109–120, 2001. doi:10.1145/603867.603884.
- 17 Fahiem Bacchus and Froduald Kabanza. Planning for temporally extended goals. In *Proceedings of the 13th AAAI conference of Artificial Intelligence*, pages 1215–1222, 1996. URL: <http://www.aaai.org/Library/AAAI/1996/aaai96-180.php>.

- 18 Marco Balduini, Alessandro Bozzon, Emanuele Della Valle, Yi Huang, and Geert-Jan Houben. Recommending venues using continuous predictive social media analytics. *IEEE Internet Comput.*, 18(5):28–35, 2014. doi:10.1109/MIC.2014.84.
- 19 Marco Balduini, Irene Celino, Daniele Dell’Aglia, Emanuele Della Valle, Yi Huang, Tony Kyung-il Lee, Seon-Ho Kim, and Volker Tresp. Reality mining on micropost streams - deductive and inductive reasoning for personalized and location-based recommendations. *Semantic Web*, 5(5):341–356, 2014. doi:10.3233/SW-130107.
- 20 Marco Balduini, Irene Celino, Daniele Dell’Aglia, Emanuele Della Valle, Yi Huang, Tony Lee, Seon-Ho Kim, and Volker Tresp. Bottari: An augmented reality mobile application to deliver personalized and location-based recommendations by continuous analysis of social media streams. *Journal of Web Semantics*, 16:33–41, 2012. doi:10.1016/j.websem.2012.06.004.
- 21 François Bancelhon. Naive evaluation of recursively defined relations. In Michael L. Brodie and John Mylopoulos, editors, *On Knowledge Base Management Systems: Integrating Artificial Intelligence and Database Technologies, Book resulting from the Islamorada Workshop 1985 (Islamorada, FL, USA)*, Topics in Information Systems, pages 165–178. Springer, 1985.
- 22 Chitta Baral, Michael Gelfond, and J. Nelson Rushton. Probabilistic reasoning with answer sets. *Theory Pract. Log. Program.*, 9(1):57–144, 2009. doi:10.1017/S1471068408003645.
- 23 Daniel Barbará. The characterization of continuous queries. *Int. J. Cooperative Inf. Syst.*, 8(4):295, 1999. doi:10.1142/S0218843099000150.
- 24 Davide Francesco Barbieri, Daniele Braga, Stefano Ceri, Emanuele Della Valle, and Michael Grossniklaus. C-sparql: a continuous query language for rdf data streams. *International Journal of Semantic Computing*, 4(01):3–25, 2010. doi:10.1142/S1793351X10000936.
- 25 Davide Francesco Barbieri, Daniele Braga, Stefano Ceri, Emanuele Della Valle, Yi Huang, Volker Tresp, Achim Rettinger, and Hendrik Wermser. Deductive and inductive stream reasoning for semantic social media analytics. *IEEE Intell. Syst.*, 25(6):32–41, 2010. doi:10.1109/MIS.2010.142.
- 26 Davide Francesco Barbieri, Daniele Braga, Stefano Ceri, Emanuele Della Valle, Yi Huang, Volker Tresp, Achim Rettinger, and Hendrik Wermser. Deductive and inductive stream reasoning for semantic social media analytics. *IEEE Intell. Syst.*, 25(6):32–41, 2010. doi:10.1109/MIS.2010.142.
- 27 Hamid R. Bazoobandi, Harald Beck, and Jacopo Urbani. Expressive stream reasoning with laser. In *ISWC*, pages 87–103, 2017. doi:10.1007/978-3-319-68288-4_6.
- 28 Harald Beck, Bruno Bierbaumer, Minh Dao-Tran, Thomas Eiter, Hermann Hellwagner, and Konstantin Schekotihin. Stream reasoning-based control of caching strategies in CCN routers. In *IEEE International Conference on Communications, ICC 2017, Paris, France, May 21-25, 2017*, pages 1–6. IEEE, 2017. doi:10.1109/ICC.2017.7996762.
- 29 Harald Beck, Minh Dao-Tran, and Thomas Eiter. LARS: A logic-based framework for analytic reasoning over streams. *Artif. Intell.*, 261:16–70, 2018. doi:10.1016/j.artint.2018.04.003.
- 30 Harald Beck, Thomas Eiter, and Christian Folie. Ticker: A system for incremental asp-based stream reasoning. *TPLP*, 17(5-6):744–763, 2017. doi:10.1017/S1471068417000370.
- 31 Alexis Bédard and Sylvain Hallé. Model checking of stream processing pipelines. In *28th International Symposium on Temporal Representation and Reasoning (TIME 2021)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.TIME.2021.5.
- 32 Matteo Belcao, Emanuele Falzone, Enea Bionda, and Emanuele Della Valle. Chimera: A bridge between big data analytics and semantic technologies. In *The Semantic Web–ISWC 2021: 20th International Semantic Web Conference, ISWC 2021, Virtual Event, October 24–28, 2021, Proceedings 20*, pages 463–479. Springer, 2021. doi:10.1007/978-3-030-88361-4_27.
- 33 Fethi Belghaoui, Amel Bouzeghoub, Zakia Kazi-Aoul, and Raja Chiky. Pol: A pattern oriented load-shedding for semantic data stream processing. In *Web Information Systems Engineering–WISE 2016: 17th International Conference, Shanghai, China, November 8-10, 2016, Proceedings, Part II 17*, pages 157–171. Springer, 2016. doi:10.1007/978-3-319-48743-4_13.
- 34 Tim Berners-Lee, James Hendler, and Ora Lassila. The semantic web: A new form of web content that is meaningful to computers will unleash a revolution of new possibilities. In Oshani Seneviratne and James A. Hendler, editors, *Linking the World’s Information: Essays on Tim Berners-Lee’s Invention of the World Wide Web*, volume 52 of *ACM Books*, pages 91–103. ACM, 2023. doi:10.1145/3591366.3591376.
- 35 Dominic Betts, Julian Dominguez, Grigori Melnik, Fernando Simonazzi, and Mani Subramanian. Exploring cqrs and event sourcing: A journey into high scalability, availability, and maintainability with windows azure, 2013. doi:10.5555/2509680.
- 36 Albert Bifet. Classifier concept drift detection and the illusion of progress. In *ICAISC (2)*, volume 10246 of *LNCS*, pages 715–725. Springer, 2017. doi:10.1007/978-3-319-59060-8_64.
- 37 Albert Bifet and Ricard Gavaldà. Adaptive Learning from Evolving Data Streams. In *IDA*, volume 5772 of *LNCS*, pages 249–260. Springer, 2009. doi:10.1007/978-3-642-03915-7_22.
- 38 Albert Bifet, Ricard Gavaldà, Geoff Holmes, and Bernhard Pfahringer. *Machine learning for data streams: with practical examples in MOA*. MIT press, 2018.
- 39 Albert Bifet, Geoff Holmes, Richard Kirkby, and Bernhard Pfahringer. MOA: massive online analysis. *J. Mach. Learn. Res.*, 11:1601–1604, 2010. doi:10.5555/1756006.1859903.
- 40 Albert Bifet, Jesse Read, Indre Zliobaite, Bernhard Pfahringer, and Geoff Holmes. Pitfalls in benchmarking data stream classification and how to avoid them. In *ECML/PKDD (1)*, volume 8188 of *Lecture Notes in Computer Science*,

- pages 465–479. Springer, 2013. doi:10.1007/978-3-642-40988-2_30.
- 41 Christian Bizer, Tom Heath, Kingsley Idehen, and Tim Berners-Lee. Linked data on the web (ldow2008). In *Proceedings of the 17th international conference on World Wide Web*, pages 1265–1266, 2008. doi:10.1145/1367497.1367760.
 - 42 Jock A Blackard and Denis J Dean. Comparative accuracies of artificial neural networks and discriminant analysis in predicting forest cover types from cartographic variables. *Computers and electronics in agriculture*, 24(3):131–151, 1999. doi:10.5555/928509.
 - 43 Andre Bolles, Marco Grawunder, and Jonas Jacobi. Streaming sparql-extending sparql to process data streams. In *The Semantic Web: Research and Applications: 5th European Semantic Web Conference, ESWC 2008, Tenerife, Canary Islands, Spain, June 1-5, 2008 Proceedings 5*, pages 448–462. Springer, 2008. doi:10.1007/978-3-540-68234-9_34.
 - 44 Pieter Bonte and Femke Ongenae. Roxi: a framework for reactive reasoning. In *The Semantic Web: ESWC 2023 Satellite Events*, pages 159–163. Springer Nature Switzerland, 2023. doi:10.1007/978-3-031-43458-7_30.
 - 45 Pieter Bonte and Femke Ongenae. Towards cascading reasoning for generic edge processing. In *ESWC2023, the First International Workshop on Semantic Web on Constrained Things*, 2023. URL: <https://ceur-ws.org/Vol-3412/paper4.pdf>.
 - 46 Pieter Bonte and Riccardo Tommasini. Streaming linked data: A survey on life cycle compliance. *Journal of Web Semantics*, 77:100785, 2023. doi:10.1016/j.websem.2023.100785.
 - 47 Pieter Bonte, Riccardo Tommasini, Filip De Turck, Femke Ongenae, and Emanuele Della Valle. C-sprite: efficient hierarchical reasoning for rapid rdf stream processing. In *Proceedings of the 13th ACM International Conference on Distributed and Event-based Systems*, pages 103–114, 2019. doi:10.1145/3328905.3329502.
 - 48 Pieter Bonte, Riccardo Tommasini, Emanuele Della Valle, Filip De Turck, and Femke Ongenae. Streaming MASSIF: Cascading reasoning for efficient processing of iot data streams. *Sensors*, 18(11):3832, 2018. doi:10.3390/s18113832.
 - 49 Stefan Borgwardt, Marcel Lippmann, and Veronika Thost. Temporalizing rewritable query languages over knowledge bases. *J. Web Semant.*, 33:50–70, 2015. doi:10.2139/ssrn.3199188.
 - 50 Irina Botan, Peter M. Fischer, Donald Kossmann, and Nesime Tatbul. Transactional stream processing. In Elke A. Rundensteiner, Volker Markl, Ioana Manolescu, Sihem Amer-Yahia, Felix Naumann, and Ismail Ari, editors, *15th International Conference on Extending Database Technology, EDBT '12, Berlin, Germany, March 27-30, 2012, Proceedings*, pages 204–215. ACM, 2012. doi:10.1145/2247596.2247622.
 - 51 Camille Bourgaux, Patrick Koopmann, and Anni-Yasmin Turhan. Ontology-mediated query answering over temporal and inconsistent data. *Semantic Web*, 10(3):475–521, 2019. doi:10.3233/SW-180337.
 - 52 Sebastian Brandt, Elem Güzel Kalayci, Vladislav Ryzhikov, Guohui Xiao, and Michael Zakharyashev. Querying log data with metric temporal logic. *J. Artif. Intell. Res.*, 62:829–877, 2018. doi:10.1613/jair.1.11229.
 - 53 Gerhard Brewka, Thomas Eiter, and Miroslaw Truszczynski. Answer set programming at a glance. *Commun. ACM*, 54(12):92–103, 2011. doi:10.1145/2043174.2043195.
 - 54 Jean-Paul Calbimonte, Davide Calvaresi, and Michael Schumacher. Multi-agent interactions on the web through linked data notifications. In *Multi-Agent Systems and Agreement Technologies: 15th European Conference, EUMAS 2017, and 5th International Conference, AT 2017, Evry, France, December 14-15, 2017, Revised Selected Papers 15*, pages 44–53. Springer, 2018. doi:10.1007/978-3-030-01713-2_4.
 - 55 Jean-Paul Calbimonte, Oscar Corcho, and Alasdair JG Gray. Enabling ontology-based access to streaming data sources. In *The Semantic Web-ISWC 2010: 9th International Semantic Web Conference, ISWC 2010, Shanghai, China, November 7-11, 2010, Revised Selected Papers, Part I 9*, pages 96–111. Springer, 2010. doi:10.1007/978-3-642-17746-0_7.
 - 56 Jean-Paul Calbimonte, Hoyoung Jeung, Oscar Corcho, and Karl Aberer. Enabling query technologies for the semantic sensor web. *International Journal On Semantic Web and Information Systems (IJSWIS)*, 8(1):43–63, 2012. doi:10.4018/jswis.2012010103.
 - 57 Jean-Paul Calbimonte, José Mora, and Óscar Corcho. Query rewriting in RDF stream processing. In *ESWC*, pages 486–502, 2016. doi:10.1007/978-3-319-34129-3_30.
 - 58 Francesco Calimeri, Marco Manna, Elena Mastria, Maria Concetta Morelli, Simona Perri, and Jessica Zangari. I-DLV-sr: A stream reasoning system based on I-DLV. *Theory Pract. Log. Program.*, 21(5):610–628, 2021. doi:10.1017/S147106842100034X.
 - 59 Paris Carbone, Marios Fragkoulis, Vasiliki Kalavri, and Asterios Katsifodimos. Beyond analytics: The evolution of stream processing systems. In *SIGMOD. ACM*, 2020. doi:10.1145/3318464.3383131.
 - 60 Paris Carbone, Asterios Katsifodimos, Stephan Ewen, Volker Markl, Seif Haridi, and Kostas Tzoumas. Apache flink: Stream and batch processing in a single engine. *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, 36(4), 2015. URL: <http://sites.computer.org/debull/A15dec/p28.pdf>.
 - 61 Stefano Ceri, Georg Gottlob, and Letizia Tanca. *Logic Programming and Databases*. Surveys in computer science. Springer, 1990. URL: <https://www.worldcat.org/oclc/20595273>.
 - 62 Sirish Chandrasekaran and Michael J. Franklin. Streaming queries over streaming data. In *VLDB*, pages 203–214. Morgan Kaufmann, 2002. doi:10.1016/B978-155860869-6/50026-3.
 - 63 Jiaoyan Chen, Freddy Lecue, Jeff Z. Pan, and Huajun Chen. Learning from ontology streams with semantic concept drift. In *Proceedings of the Twenty-Sixth International Joint Conference on*

- Artificial Intelligence, IJCAI-17*, pages 957–963, 2017. doi:10.24963/ijcai.2017/133.
- 64 Michael Compton, Payam Barnaghi, Luis Bermudez, Raul Garcia-Castro, Oscar Corcho, Simon Cox, John Graybeal, Manfred Hauswirth, Cory Henson, Arthur Herzog, et al. The ssn ontology of the w3c semantic sensor network incubator group. *Journal of Web Semantics*, 17:25–32, 2012. doi:10.1016/j.websem.2012.05.003.
- 65 Simon Cox and Chris Little. Time ontology in OWL. Technical report, W3C, Spatial Data on the Web Working Group, 2022. W3C Candidate Recommendation Draft, Nov 25, 2022, <https://www.w3.org/TR/owl-time/>. URL: <https://www.w3.org/TR/owl-time/>.
- 66 David J. Tena Cucala, Przemyslaw Andrzej Walega, Bernardo Cuenca Grau, and Egor V. Kostylev. Stratified negation in datalog with metric temporal operators. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pages 6488–6495. AAAI Press, 2021. doi:10.1609/AAAI.V35I7.16804.
- 67 Gianpaolo Cugola and Alessandro Margara. Processing flows of information: From data stream to complex event processing. *ACM Comput. Surv.*, 44(3):15:1–15:62, 2012. doi:10.1145/2187671.2187677.
- 68 Gianpaolo Cugola, Alessandro Margara, Matteo Matteucci, and Giordano Tamburrelli. Introducing uncertainty in complex event processing: model, implementation, and validation. *Computing*, 97(2):103–144, 2015. doi:10.1007/s00607-014-0404-y.
- 69 Richard Cyganiak, David Wood, and Markus Lanthaler. RDF 1.1 Concepts and Abstract Syntax. W3C Recommendation, W3C, 2014. URL: <https://www.w3.org/TR/rdf11-concepts/>.
- 70 Ariyam Das, Sahil M. Gandhi, and Carlo Zaniolo. ASTRO: A datalog system for advanced stream reasoning. In *CIKM*, pages 1863–1866, 2018. doi:10.1145/3269206.3269223.
- 71 Mathias De Brouwer, Pieter Bonte, Dörthe Arndt, Miel Vander Sande, Pieter Heyvaert, Anastasia Dimou, Ruben Verborgh, Filip De Turck, and Femke Ongenaë. Distributed continuous home care provisioning through personalized monitoring & treatment planning. In *Companion Proceedings of the Web Conference 2020*, pages 143–147, 2020. doi:10.1145/3366424.3383528.
- 72 Daniel de Leng and Fredrik Heintz. DyKnow: A dynamically reconfigurable stream reasoning framework as an extension to the robot operating system. In *SIMPAR*, pages 55–60. IEEE, 2016. doi:10.1109/SIMPAR.2016.7862375.
- 73 Daniel de Leng and Fredrik Heintz. Partial-state progression for stream reasoning with metric temporal logic. In *Sixteenth International Conference on Principles of Knowledge Representation and Reasoning*, 2018. URL: <https://aaai.org/ocs/index.php/KR/KR18/paper/view/17988>.
- 74 Daniel de Leng and Fredrik Heintz. Approximate stream reasoning with metric temporal logic under uncertainty. In *AAAI*, pages 2760–2767, 2019. doi:10.1609/aaai.v33i01.33012760.
- 75 Emanuele Della Valle. *On Stream Reasoning*. PhD Thesis, Vrije Universiteit Amsterdam, 2015. URL: <https://research.vu.nl/en/publications/on-stream-reasoning>.
- 76 Emanuele Della Valle, Irene Celino, Daniele Dell’Aglío, Ralph Grothmann, Florian Steinke, and Volker Tresp. Semantic traffic-aware routing using the larkc platform. *IEEE Internet Comput.*, 15(6):15–23, 2011. doi:10.1109/MIC.2011.107.
- 77 Daniele Dell’Aglío and Abraham Bernstein. Differentially private stream processing for the semantic web. In *WWW*, pages 1977–1987. ACM / IW3C2, 2020. doi:10.1145/3366423.3380265.
- 78 Daniele Dell’Aglío, Emanuele Della Valle, Jean-Paul Calbimonte, and Óscar Corcho. RSP-QL semantics: A unifying query model to explain heterogeneity of RDF stream processing systems. *Int. J. Semantic Web Inf. Syst.*, 10(4):17–44, 2014. doi:10.4018/ijswis.2014100102.
- 79 Daniele Dell’Aglío, Emanuele Della Valle, et al. Incremental reasoning on rdf streams, 2014. doi:10.1201/B16859-22.
- 80 Daniele Dell’Aglío, Danh Le Phuoc, Anh Le-Tuan, Muhammad Intizar Ali, and Jean-Paul Calbimonte. On a web of data streams. In *Proceedings of the workshop on decentralizing the semantic Web 2017 co-located with 16th International Semantic Web Conference (ISWC 2017)*. 22 October 2017, 2017. URL: <https://ceur-ws.org/Vol-1934/contribution-11.pdf>.
- 81 Daniele Dell’Aglío, Jean-Paul Calbimonte, Marco Balduini, Oscar Corcho, and Emanuele Della Valle. On correctness in rdf stream processor benchmarking. In *The Semantic Web–ISWC 2013: 12th International Semantic Web Conference, Sydney, NSW, Australia, October 21-25, 2013, Proceedings, Part II 12*, pages 326–342. Springer, 2013. doi:10.1007/978-3-642-41338-4_21.
- 82 Daniele Dell’Aglío, Minh Dao-Tran, Jean-Paul Calbimonte, Danh Le Phuoc, and Emanuele Della Valle. A query model to capture event pattern matching in rdf stream processing query languages. In *European Knowledge Acquisition Workshop*, pages 145–162. Springer, 2016. doi:10.1007/978-3-319-49004-5_10.
- 83 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. URL: https://ceur-ws.org/Vol-1184/ldow2014_paper_01.pdf.
- 84 Gregory Ditzler and Robi Polikar. Incremental Learning of Concept Drift from Streaming Imbalanced Data. *IEEE Trans. Knowl. Data Eng.*, 25(10):2283–2301, 2013. doi:10.1109/TKDE.2012.136.
- 85 Patrick Doherty and Jonas Kvarnström. Temporal action logics. In Frank van Harmelen, Vladimir Lifschitz, and Bruce W. Porter, editors, *Handbook of Knowledge Representation*, volume 3 of *Foundations of Artificial Intelligence*, pages 709–757. Elsevier, 2008. doi:10.1016/S1574-6526(07)03018-0.

- 86 Patrick Doherty, Jonas Kvarnström, and Fredrik Heintz. A temporal logic-based planning and execution monitoring framework for unmanned aircraft systems. *Auton. Agent Multi-Ag.*, 19(3):332–377, 2009. doi:10.1007/s10458-009-9079-8.
- 87 Manh Nguyen Duc, Anh Lê Tuán, Manfred Hauswirth, and Danh Le Phuoc. Towards autonomous semantic stream fusion for distributed video streams. In Alessandro Margara, Emanuele Della Valle, Alexander Artikis, Nesime Tatbul, and Helge Parzyjeglá, editors, *15th ACM International Conference on Distributed and Event-based Systems, DEBS 2021, Virtual Event, Italy, June 28 - July 2, 2021*, pages 172–175. ACM, 2021. doi:10.1145/3465480.3467837.
- 88 Martin Dürst and Michel Suignard. Internationalized resource identifiers (IRIs). Technical report, W3C, Internationalization Working Group, 2005. doi:10.17487/RFC3987.
- 89 Thomas Eiter, Ryutaro Ichise, Josiane Xavier Parreira, Patrik Schneider, and Lihua Zhao. Deploying spatial-stream query answering in C-ITS scenarios. *Semantic Web*, 12(1):41–77, 2021. doi:10.3233/SW-200408.
- 90 Thomas Eiter and Rafael Kiesel. Weighted LARS for quantitative stream reasoning. In *ECAI*, pages 729–736, 2020. doi:10.3233/FAIA200160.
- 91 Thomas Eiter and Rafael Kiesel. Semiring reasoning frameworks in AI and their computational complexity. *J. Artif. Intell. Res.*, 77:207–293, 2023. doi:10.1613/jair.1.13970.
- 92 Thomas Eiter, Paul Ogris, and Konstantin Schekotihin. A distributed approach to LARS stream reasoning (system paper). *TPLP*, 19(5-6):974–989, 2019. doi:10.1017/S1471068419000309.
- 93 Emanuele Falzone, Riccardo Tommasini, Emanuele Della Valle, Petra Selmer, Stefan Plantikow, Hannes Voigt, Keith Hare, Ljubica Lazarevic, and Tobias Lindaaker. Semantic foundations of seraph continuous graph query language. *arXiv preprint arXiv:2111.09228*, 2021. doi:10.48550/arXiv.2111.09228.
- 94 Javier D Fernández, Alejandro Llavés, and Oscar Corcho. Efficient rdf interchange (eri) format for rdf data streams. In *The Semantic Web-ISWC 2014: 13th International Semantic Web Conference, Riva del Garda, Italy, October 19-23, 2014. Proceedings, Part II 13*, pages 244–259. Springer, 2014. doi:10.1007/978-3-319-11915-1_16.
- 95 Michael Fisher. Temporal representation and reasoning. In Frank van Harmelen, Vladimir Lifschitz, and Bruce W. Porter, editors, *Handbook of Knowledge Representation*, volume 3 of *Foundations of Artificial Intelligence*, pages 513–550. Elsevier, 2008. doi:10.1016/S1574-6526(07)03012-X.
- 96 Michael Fisher, Dov M. Gabbay, and Lluís Vila, editors. *Handbook of Temporal Reasoning in Artificial Intelligence*, volume 1 of *Foundations of Artificial Intelligence*. Elsevier, 2005. doi:10.5555/2974992.
- 97 João Gama, Pedro Medas, Gladys Castillo, and Pedro Pereira Rodrigues. Learning with Drift Detection. In *SBIA*, volume 3171 of *LNCIS*, pages 286–295. Springer, 2004. doi:10.1007/978-3-540-28645-5_29.
- 98 João Gama, Raquel Sebastião, and Pedro Pereira Rodrigues. Issues in evaluation of stream learning algorithms. In *KDD*, pages 329–338. ACM, 2009. doi:10.1145/1557019.1557060.
- 99 Jing Gao, Wei Fan, Jiawei Han, and Philip Yu. A general framework for mining concept-drifting data streams with skewed distributions. In *Proceedings of the 2007 SIAM international conference on data mining*, pages 3–14, apr 2007. doi:10.1137/1.9781611972771.1.
- 100 Shen Gao, Thomas Scharrenbach, and Abraham Bernstein. The clock data-aware eviction approach: Towards processing linked data streams with limited resources. In *European Semantic Web Conference*, pages 6–20. Springer, 2014. doi:10.1007/978-3-319-07443-6_2.
- 101 Martin Gebser, Roland Kaminski, Benjamin Kaufmann, and Torsten Schaub. Multi-shot ASP solving with clingo. *TPLP*, 19(1):27–82, 2019. doi:10.1017/S1471068418000054.
- 102 Stefano Germano, Thu-Le Pham, and Alessandra Mileo. Web stream reasoning in practice: On the expressivity vs. scalability tradeoff. In Balder ten Cate and Alessandra Mileo, editors, *Web Reasoning and Rule Systems - 9th International Conference, RR 2015, Berlin, Germany, August 4-5, 2015, Proceedings*, volume 9209 of *Lecture Notes in Computer Science*, pages 105–112. Springer, 2015. doi:10.1007/978-3-319-22002-4_9.
- 103 Federico Giannini, Giacomo Ziffer, and Emanuele Della Valle. cpnn: Continuous progressive neural networks for evolving streaming time series. In Hisashi Kashima, Tsuyoshi Idé, and Wen-Chih Peng, editors, *Advances in Knowledge Discovery and Data Mining - 27th Pacific-Asia Conference on Knowledge Discovery and Data Mining, PAKDD 2023, Osaka, Japan, May 25-28, 2023, Proceedings, Part IV*, volume 13938 of *Lecture Notes in Computer Science*, pages 328–340. Springer, 2023. doi:10.1007/978-3-031-33383-5_26.
- 104 Nikos Giatrakos, Alexander Artikis, Antonios Deligiannakis, and Minos N. Garofalakis. Complex event recognition in the big data era. *Proc. VLDB Endow.*, 10(12):1996–1999, 2017. doi:10.14778/3137765.3137829.
- 105 Boris Glavic, Kyumars Sheykh Esmaili, Peter M. Fischer, and Nesime Tatbul. Efficient stream provenance via operator instrumentation. *ACM Trans. Internet Techn.*, 14(1):7:1–7:26, 2014. doi:10.1145/2633689.
- 106 Boris Glavic, Kyumars Sheykh Esmaili, Peter Michael Fischer, and Nesime Tatbul. Ariadne: managing fine-grained provenance on data streams. In Sharma Chakravarthy, Susan Darling Urban, Peter R. Pietzuch, and Elke A. Rundensteiner, editors, *The 7th ACM International Conference on Distributed Event-Based Systems, DEBS '13, Arlington, TX, USA - June 29 - July 03, 2013*, pages 39–50. ACM, 2013. doi:10.1145/2488222.2488256.
- 107 Heitor Murilo Gomes, Albert Bifet, Jesse Read, Jean Paul Barddal, Fabrício Enembreck, Bernhard Pfahringer, Geoff Holmes, and Talel Abdesslem. Adaptive random forests for evolving data stream

- classification. *Mach. Learn.*, 106(9-10):1469–1495, 2017. doi:10.1007/s10994-017-5642-8.
- 108 Ian J. Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio. An Empirical Investigation of Catastrophic Forgetting in Gradient-Based Neural Networks. In *2nd International Conference on Learning Representations*, 2015-03-03. doi:10.48550/arXiv.1312.6211.
- 109 Felipe Gorostiaga and César Sánchez. Hstriver: a very functional extensible tool for the runtime verification of real-time event streams. In *International Symposium on Formal Methods*, pages 563–580. Springer, 2021. doi:10.1007/978-3-030-90870-6_30.
- 110 OWL Working group. OWL 2 web ontology language overview (second edition). Technical report, W3C, dec 2012. W3C recommendation. URL: <http://www.w3.org/TR/2012/REC-owl2-overview-20121211/>.
- 111 Aakansha Gupta and Rahul Katarya. Social media based surveillance systems for healthcare using machine learning: A systematic review. *J. Biomed. Inform.*, 108:103500, 2020. doi:10.1016/j.jbi.2020.103500.
- 112 Fredrik Heintz, Jonas Kvarnström, and Patrick Doherty. Bridging the sense-reasoning gap: Dknow - stream-based middleware for knowledge processing. *Adv. Eng. Inform.*, 24(1):14–26, 2010. doi:10.1016/j.aei.2009.08.007.
- 113 Fredrik Heintz, Jonas Kvarnström, and Patrick Doherty. Stream-based reasoning support for autonomous systems. In Helder Coelho, Rudi Studer, and Michael J. Wooldridge, editors, *ECAI 2010 - 19th European Conference on Artificial Intelligence, Lisbon, Portugal, August 16-20, 2010, Proceedings*, volume 215 of *Frontiers in Artificial Intelligence and Applications*, pages 183–188. IOS Press, 2010. doi:10.3233/978-1-60750-606-5-183.
- 114 Pieter Heyvaert, Ben De Meester, Anastasia Dimou, and Ruben Verborgh. Declarative rules for linked data generation at your fingertips! In *European Semantic Web Conference*, pages 213–217. Springer, 2018. doi:10.1007/978-3-319-98192-5_40.
- 115 Martin Hirzel, Guillaume Baudart, Angela Bonifati, Emanuele Della Valle, Sherif Sakr, and Akrivi Vlachou. Stream processing languages in the big data era. *SIGMOD Record*, 47(2):29–40, 2018. doi:10.1145/3299887.3299892.
- 116 Ian M. Hodkinson and Mark Reynolds. Temporal logic. In Patrick Blackburn, J. F. A. K. van Benthem, and Frank Wolter, editors, *Handbook of Modal Logic*, volume 3 of *Studies in logic and practical reasoning*, pages 655–720. North-Holland, 2007. doi:10.1016/s1570-2464(07)80014-0.
- 117 Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *The collected works of Wassily Hoeffding*, pages 409–426, 1994. doi:10.1007/978-1-4612-0865-5_26.
- 118 Pan Hu, Boris Motik, and Ian Horrocks. Modular materialisation of Datalog programs. *Artif. Intell.*, 308:103726, 2022. doi:10.1016/j.artint.2022.103726.
- 119 Giovambattista Ianni, Francesco Pacenza, and Jessica Zangari. Incremental maintenance of overgrounded logic programs with tailored simplifications. *Theory Pract. Log. Program.*, 20(5):719–734, 2020. doi:10.1017/S147106842000040X.
- 120 Elena Ikonomovska, João Gama, and Saso Dzeroski. Learning model trees from evolving data streams. *Data Min. Knowl. Discov.*, 23(1):128–168, 2011. doi:10.1007/s10618-010-0201-y.
- 121 Neil Immerman. *Descriptive complexity*. Graduate texts in computer science. Springer, 1999. doi:10.1007/978-1-4612-0539-5.
- 122 Sebastian Käbisch, Daniel Peintner, and Darko Anicic. Standardized and efficient rdf encoding for constrained embedded networks. In *European Semantic Web Conference*, pages 437–452. Springer, 2015. doi:10.1007/978-3-319-18818-8_27.
- 123 Antonis C. Kakas. Abduction. In Claude Sammut and Geoffrey I. Webb, editors, *Encyclopedia of Machine Learning*, pages 3–9. Springer, 2010. doi:10.1007/978-0-387-30164-8_1.
- 124 Andreas Kamilaris, Feng Gao, Francesc X. Prenafeta-Boldu, and Muhammad Intizar Ali. Agri-iot: A semantic framework for internet of things-enabled smart farming applications. In *3rd IEEE World Forum on Internet of Things, WF-IoT 2016, Reston, VA, USA, December 12-14, 2016*, pages 442–447. IEEE Computer Society, 2016. doi:10.1109/WF-IoT.2016.7845467.
- 125 Evgeny Kharlamov, Yannis Kotidis, Theofilos Mailis, Christian Neuenstadt, Charalampos Nikolaou, Özgür L. Özçep, Christoforos Svingos, Dmitriy Zheleznyakov, Sebastian Brandt, Ian Horrocks, Yannis E. Ioannidis, Steffen Lamparter, and Ralf Möller. Towards analytics aware ontology based access to static and streaming data. In *ISWC (2)*, volume 9982 of *Lecture Notes in Computer Science*, pages 344–362, 2016. doi:10.1007/978-3-319-46547-0_31.
- 126 Houda Khrouf, Badre Belabbess, Laurent Bihanic, Gabriel Kepekian, and Olivier Curé. Waves: Big data platform for real-time rdf stream processing. In Daniele Dell’Aglio, Emanuele Della Valle, Markus Krötzsch, Thomas Eiter, Maria Maleshkova, Ruben Verborgh, Federico M. Facca, and Michael Mrissa, editors, *Joint Proceedings of the 3rd Stream Reasoning (SR 2016) and the 1st Semantic Web Technologies for the Internet of Thing (SWIT 2016) workshops (SR+SWIT)*, number 1783 in CEUR Workshop Proceedings, pages 37–48, Aachen, 2016. URL: <https://ceur-ws.org/Vol-1783/paper-04.pdf>.
- 127 James Kirkpatrick, Razvan Pascanu, Neil C. Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13), 2017. doi:10.1073/pnas.1611835114.
- 128 Maxim Kolchin, Peter Wetz, Elmar Kiesling, and A Min Tjoa. Yabench: A comprehensive framework for rdf stream processor correctness and performance assessment. In *Web Engineering: 16th International Conference, ICWE 2016*,

- Lugano, Switzerland, June 6-9, 2016. *Proceedings 16*, pages 280–298. Springer, 2016. doi:10.1007/978-3-319-38791-8_16.
- 129 Nicolas Kourtellis, Gianmarco De Francisci Morales, Albert Bifet, and Arinto Murdopo. VHT: vertical hoeffding tree. In James Joshi, George Karypis, Ling Liu, Xiaohua Hu, Ronay Ak, Yinglong Xia, Weijia Xu, Aki-Hiro Sato, Sudarsan Rachuri, Lyle H. Ungar, Philip S. Yu, Rama Govindaraju, and Toyotaro Suzumura, editors, *2016 IEEE International Conference on Big Data (IEEE BigData 2016), Washington DC, USA, December 5-8, 2016*, pages 915–922. IEEE Computer Society, 2016. doi:10.1109/BIGDATA.2016.7840687.
 - 130 Ron Koymans. Specifying real-time properties with metric temporal logic. *Real-Time Syst.*, 2(4):255–299, 1990. doi:10.1007/BF01995674.
 - 131 Jeffrey R Lacasse and Eileen Gambrill. Making assessment decisions: Macro, mezzo, and micro perspectives. *Critical thinking in clinical assessment and diagnosis*, pages 69–84, 2015. doi:10.1007/978-3-319-17774-8_4.
 - 132 Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Ales Leonardis, Gregory G. Slabaugh, and Tinne Tuytelaars. A Continual Learning Survey: Defying Forgetting in Classification Tasks. *IEEE Trans. Pattern Anal. Mach. Intell.*, 44(7):3366–3385, 2022. doi:10.1109/TPAMI.2021.3057446.
 - 133 Danh Le-Phuoc, Minh Dao-Tran, Minh-Duc Pham, Peter Boncz, Thomas Eiter, and Michael Fink. Linked stream data processing engines: Facts and figures. In *International Semantic Web Conference*, pages 300–312. Springer, 2012. doi:10.1007/978-3-642-35173-0_20.
 - 134 Danh Le-Phuoc, Minh Dao-Tran, Josiane Xavier Parreira, and Manfred Hauswirth. A native and adaptive approach for unified processing of linked streams and linked data. In *International Semantic Web Conference*, pages 370–388. Springer, 2011. doi:10.1007/978-3-642-25073-6_24.
 - 135 Danh Le-Phuoc, Hoan Nguyen Mau Quoc, Chan Le Van, and Manfred Hauswirth. Elastic and scalable processing of linked stream data in the cloud. In *International Semantic Web Conference*, pages 280–297. Springer, 2013. doi:10.1007/978-3-642-41335-3_18.
 - 136 Anh Le-Tuan, Manh Nguyen-Duc, Chien-Quang Le, Trung-Kien Tran, Manfred Hauswirth, Thomas Eiter, and Danh Le-Phuoc. Cqels 2.0: Towards a unified framework for semantic stream fusion. *arXiv preprint arXiv:2202.13958*, 2022. doi:10.48550/arXiv.2202.13958.
 - 137 Freddy Lécué. Towards scalable exploration of diagnoses in an ontology stream. In Carla E. Brodley and Peter Stone, editors, *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada*, pages 87–93. AAAI Press, 2014. doi:10.1609/aaai.v28i1.8708.
 - 138 Freddy Lécué and Jeff Z. Pan. Predicting knowledge in an ontology stream. In *IJCAI*, pages 2662–2669. IJCAI/AAAI, 2013. doi:10.5555/2540128.2540512.
 - 139 Freddy Lécué, Simone Tallevi-Diotalleivi, Jer Hayes, Robert Tucker, Veli Bicer, Marco Luca Sbodio, and Pierpaolo Tommasi. Smart traffic analytics in the semantic web with STAR-CITY: scenarios, system and lessons learned in dublin city. *J. Web Semant.*, 27-28:26–33, 2014. doi:10.1016/j.websem.2014.07.002.
 - 140 Maurizio Lenzerini. Data integration: A theoretical perspective. In Lucian Popa, Serge Abiteboul, and Phokion G. Kolaitis, editors, *Proceedings of the Twenty-first ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 3-5, Madison, Wisconsin, USA*, pages 233–246. ACM, 2002. doi:10.1145/543613.543644.
 - 141 Timothée Lesort, Vincenzo Lomonaco, Andrei Stoian, Davide Maltoni, David Filliat, and Natalia Díaz Rodríguez. Continual learning for robotics: Definition, framework, learning strategies, opportunities and challenges. *Inf. Fusion*, 58:52–68, 2020. doi:10.1016/j.inffus.2019.12.004.
 - 142 Zhizhong Li and Derek Hoiem. Learning Without Forgetting. In *ECCV (4)*, volume 9908 of *Lecture Notes in Computer Science*, pages 614–629. Springer, 2016. doi:10.1007/978-3-319-46493-0_37.
 - 143 Zhiqiu Lin, Jia Shi, Deepak Pathak, and Deva Ramanan. The CLEAR benchmark: Continual learning on real-world imagery. In Joaquin Vanschoren and Sai-Kit Yeung, editors, *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*, 2021. URL: <https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/hash/2838023a778dfaecd212708f721b788-Abstract-round2.html>.
 - 144 Mo Liu, Ming Li, Denis Golovnya, Elke A. Rundensteiner, and Kajal T. Claypool. Sequence pattern query processing over out-of-order event streams. In Yannis E. Ioannidis, Dik Lun Lee, and Raymond T. Ng, editors, *Proceedings of the 25th International Conference on Data Engineering, ICDE 2009, March 29 2009 - April 2 2009, Shanghai, China*, pages 784–795. IEEE Computer Society, 2009. doi:10.1109/ICDE.2009.95.
 - 145 Vincenzo Lomonaco, Davide Maltoni, and Lorenzo Pellegrini. Rehearsal-Free Continual Learning over Small Non-I.I.D. Batches. In *CVPR Workshops*, pages 989–998. Computer Vision Foundation / IEEE, 2020. doi:10.1109/CVPRW50498.2020.00131.
 - 146 Vincenzo Lomonaco, Lorenzo Pellegrini, Andrea Cossu, Antonio Carta, Gabriele Graffieti, Tyler L. Hayes, Matthias De Lange, Marc Masana, Jary Pomponi, Gido van de Ven, Martin Mundt, Qi She, Keiland Cooper, Jeremy Forest, Eden Belouadah, Simone Calderara, German I. Parisi, Fabio Cuzzolin, Andreas Tolia, Simone Scardapane, Luca Antiga, Subutai Amhad, Adrian Popescu, Christopher Kanan, Joost van de Weijer, Tinne Tuytelaars, Davide Bacciu, and Davide Maltoni. Avalanche: an end-to-end library for continual learning. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2nd Con-

- tinual Learning in Computer Vision Workshop, 2021. doi:10.1109/CVPRW53098.2021.00399.
- 147 David Lopez-Paz and Marc'Aurelio Ranzato. Gradient Episodic Memory for Continual Learning. In *NIPS*, pages 6467–6476, 2017. doi:10.5555/3295222.3295393.
 - 148 Jie Lu, Anjin Liu, Fan Dong, Feng Gu, João Gama, and Guangquan Zhang. Learning under Concept Drift: A Review. *IEEE Trans. Knowl. Data Eng.*, 31(12):2346–2363, 2019. doi:10.1109/TKDE.2018.2876857.
 - 149 Robin Manhaeve, Sebastijan Dumancic, Angelika Kimmig, Thomas Demeester, and Luc De Raedt. Neural probabilistic logic programming in deep-probrog. *Artif. Intell.*, 298:103504, 2021. doi:10.1016/j.artint.2021.103504.
 - 150 Andrea Mauri, Jean-Paul Calbimonte, Daniele Dell'Aglia, Marco Balduini, Marco Brambilla, Emanuele Della Valle, and Karl Aberer. Triple-Wave: Spreading RDF Streams on the Web. In *International Semantic Web Conference (2)*, volume 9982 of *Lecture Notes in Computer Science*, pages 140–149. Springer, 2016. doi:10.1007/978-3-319-46547-0_15.
 - 151 Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier, 1989. doi:10.1016/S0079-7421(08)60536-8.
 - 152 Alessandra Mileo, Ahmed Abdelrahman, Sean Polcarpio, and Manfred Hauswirth. Streamrule: A nonmonotonic stream reasoning system for the semantic web. In *RR*, pages 247–252, 2013. doi:10.1007/978-3-642-39666-3_23.
 - 153 Alessandra Mileo, Minh Dao-Tran, Thomas Eiter, and Michael Fink. Stream reasoning. In Ling Liu and M. Tamer Özsu, editors, *Encyclopedia of Database Systems, Second Edition*. Springer, 2018. doi:10.1007/978-1-4614-8265-9_80715.
 - 154 Jacob Montiel, Max Halford, Saulo Martiello Mastelini, Geoffrey Bolmier, Raphaël Sourty, Robin Vaysse, Adil Zouitine, Heitor Murilo Gomes, Jesse Read, Talel Abdesslem, and Albert Bifet. River: machine learning for streaming data in python. *J. Mach. Learn. Res.*, 22:110:1–110:8, 2021. URL: <http://jmlr.org/papers/v22/20-1380.html>.
 - 155 Gianmarco De Francisci Morales and Albert Bifet. SAMOA: scalable advanced massive online analysis. *J. Mach. Learn. Res.*, 16:149–153, 2015. doi:10.5555/2789272.2789277.
 - 156 Boris Motik, Yavor Nenov, Robert Piro, and Ian Horrocks. Maintenance of datalog materialisations revisited. *Artif. Intell.*, 269:76–136, 2019. doi:10.1016/j.artint.2018.12.004.
 - 157 Boris Motik, Yavor Nenov, Robert Piro, Ian Horrocks, and Dan Olteanu. Parallel Materialisation of Datalog Programs in Centralised, Main-Memory RDF Systems. In Carla E. Brodley and Peter Stone, editors, *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada*, pages 129–137. AAAI Press, 2014. doi:10.1609/aaai.v28i1.8730.
 - 158 Esteban Municio, Glenn Daneels, Mathias De Brouwer, Femke Ongenaes, Filip De Turck, Bart Braem, Jeroen Famaey, and Steven Latré. Continuous athlete monitoring in challenging cycling environments using iot technologies. *IEEE Internet of Things Journal*, 6(6):10875–10887, 2019. doi:10.1109/JIOT.2019.2942761.
 - 159 S. Muthukrishnan. Data streams: Algorithms and applications. *Found. Trends Theor. Comput. Sci.*, 1(2), 2005. doi:10.1561/0400000002.
 - 160 Yavor Nenov, Robert Piro, Boris Motik, Ian Horrocks, Zhe Wu, and Jay Banerjee. RD-Fox: A Highly-Scalable RDF Store. In Marcelo Arenas, Óscar Corcho, Elena Simperl, Markus Strohmaier, Mathieu d'Aquin, Kavitha Srinivas, Paul Groth, Michel Dumontier, Jeff Heflin, Krishnaprasad Thirunarayan, and Steffen Staab, editors, *The Semantic Web - ISWC 2015 - 14th International Semantic Web Conference, Bethlehem, PA, USA, October 11-15, 2015, Proceedings, Part II*, volume 9367 of *Lecture Notes in Computer Science*, pages 3–20. Springer, 2015. doi:10.1007/978-3-319-25010-6_1.
 - 161 Matthias Nickles and Alessandra Mileo. Web stream reasoning using probabilistic answer set programming. In Roman Kontchakov and Marie-Laure Mugnier, editors, *Web Reasoning and Rule Systems - 8th International Conference, RR 2014, Athens, Greece, September 15-17, 2014. Proceedings*, volume 8741 of *Lecture Notes in Computer Science*, pages 197–205. Springer, 2014. doi:10.1007/978-3-319-11113-1_16.
 - 162 Matthias Nickles and Alessandra Mileo. A hybrid approach to inference in probabilistic non-monotonic logic programming. In Fabrizio Riguzzi and Joost Vennekens, editors, *Proceedings of the 2nd International Workshop on Probabilistic Logic Programming co-located with 31st International Conference on Logic Programming (ICLP 2015), Cork, Ireland, August 31st, 2015*, volume 1413 of *CEUR Workshop Proceedings*, pages 57–68. CEUR-WS.org, 2015. URL: <https://ceur-ws.org/Vol-1413/paper-05.pdf>.
 - 163 Matthias Nickles and Alessandra Mileo. A system for probabilistic inductive answer set programming. In Christoph Beierle and Alex Dekhtyar, editors, *Scalable Uncertainty Management - 9th International Conference, SUM 2015, Québec City, QC, Canada, September 16-18, 2015. Proceedings*, volume 9310 of *Lecture Notes in Computer Science*, pages 99–105. Springer, 2015. doi:10.1007/978-3-319-23540-0_7.
 - 164 Philipp Obermeier, Javier Romero, and Torsten Schaub. Multi-shot stream reasoning in answer set programming: A preliminary report. *OJDB*, 6(1):33–38, 2019. URL: https://www.ronpub.com/ojdb/OJDB_2019v6i1n04_Obermeier.html.
 - 165 Michiel Overeem, Marten Spoor, and Slinger Jansen. The dark side of event sourcing: Managing data conversion. In *2017 IEEE 24th international conference on software analysis, evolution and reengineering (SANER)*, pages 193–204. IEEE, 2017. doi:10.1109/SANER.2017.7884621.
 - 166 Özgür Lütü Özçep, Ralf Möller, and Christian Neuenstadt. A stream-temporal query language for ontology based data access. In *KI 2014: Advances in Artificial Intelligence: 37th Annual German Conference on AI, Stuttgart, Ger-*




- many, September 22–26, 2014. *Proceedings 37*, pages 183–194. Springer, 2014. doi:10.1007/978-3-319-11206-0_18.
- 167 Özgür Lütfü Özçep, Ralf Möller, and Christian Neuenstadt. Stream-query compilation with ontologies. In Bernhard Pfahringer and Jochen Renz, editors, *AI 2015: Advances in Artificial Intelligence - 28th Australasian Joint Conference, Canberra, ACT, Australia, November 30 - December 4, 2015, Proceedings*, volume 9457 of *Lecture Notes in Computer Science*, pages 457–463. Springer, 2015. doi:10.1007/978-3-319-26350-2_40.
- 168 Anil Pacaci, Angela Bonifati, and M. Tamer Özsu. Regular path query evaluation on streaming graphs. In David Maier, Rachel Pottinger, An-Hai Doan, Wang-Chiew Tan, Abdussalam Alawini, and Hung Q. Ngo, editors, *Proceedings of the 2020 International Conference on Management of Data, SIGMOD Conference 2020, online conference [Portland, OR, USA], June 14–19, 2020*, pages 1415–1430. ACM, 2020. doi:10.1145/3318464.3389733.
- 169 Anil Pacaci, Angela Bonifati, and M. Tamer Özsu. Evaluating complex queries on streaming graphs. In *38th IEEE International Conference on Data Engineering, ICDE 2022, Kuala Lumpur, Malaysia, May 9–12, 2022*, pages 272–285. IEEE, 2022. doi:10.1109/ICDE53745.2022.00025.
- 170 Dimitris Palyvos-Giannas, Vincenzo Gulisano, and Marina Papatriantafidou. Genealog: Fine-grained data streaming provenance at the edge. In *Proceedings of the 19th International Middleware Conference*, pages 227–238, 2018. doi:10.1145/3274808.3274826.
- 171 Dimitris Palyvos-Giannas, Bastian Havers, Marina Papatriantafidou, and Vincenzo Gulisano. Ananke: A streaming framework for live forward provenance. *Proc. VLDB Endow.*, 14(3):391–403, 2020. doi:10.5555/3430915.3442437.
- 172 Dimitris Palyvos-Giannas, Katerina Tzompanaki, Marina Papatriantafidou, and Vincenzo Gulisano. Erebus: Explaining the outputs of data streaming queries. *Proc. VLDB Endow.*, 16(2):230–242, 2022. doi:10.14778/3565816.3565825.
- 173 Douglas S. Parker. Integrating AI and DBMS through stream processing. In *ICDE*, pages 259–260, 1989. doi:10.1109/ICDE.1989.47224.
- 174 Stott D. Parker. Stream data analysis in prolog. In *The Practice of Prolog*, pages 249–301. MIT Press, 2003.
- 175 Harshal Patni, Cory Henson, and Amit Sheth. Linked sensor data. In *2010 International Symposium on Collaborative Technologies and Systems*, pages 362–370. IEEE, 2010. doi:10.1109/CTS.2010.5478492.
- 176 Lorenzo Pellegrini, Gabriele Graffieti, Vincenzo Lomonaco, and Davide Maltoni. Latent Replay for Real-Time Continual Learning. In *IROS*, pages 10203–10209. IEEE, 2020. doi:10.1109/IROS45743.2020.9341460.
- 177 Romana Pernisch, Daniele Dell’Aglia, and Abraham Bernstein. Beware of the hierarchy — An analysis of ontology evolution and the materialisation impact for biomedical ontologies. *Journal of Web Semantics*, 70:100658, jul 2021. doi:10.1016/j.websem.2021.100658.
- 178 Thu-Le Pham, Muhammad Intizar Ali, and Alessandra Mileo. C-ASP: Continuous ASP-based reasoning over RDF streams. In *LPNMR*, pages 45–50, 2019. doi:10.1007/978-3-030-20528-7_4.
- 179 Thu-Le Pham, Muhammad Intizar Ali, and Alessandra Mileo. Enhancing the scalability of expressive stream reasoning via input-driven parallelization. *Semantic Web*, 10(3):457–474, 2019. doi:10.3233/SW-180330.
- 180 Thu-Le Pham, Alessandra Mileo, and Muhammad Intizar Ali. Towards scalable non-monotonic stream reasoning via input dependency analysis. In *33rd IEEE International Conference on Data Engineering, ICDE 2017, San Diego, CA, USA, April 19–22, 2017*, pages 1553–1558. IEEE Computer Society, 2017. doi:10.1109/ICDE.2017.226.
- 181 Danh Le Phuoc, Thomas Eiter, and Anh Lê Tuán. A scalable reasoning and learning approach for neural-symbolic stream fusion. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2–9, 2021*, pages 4996–5005. AAAI Press, 2021. doi:10.1609/aaai.v35i6.16633.
- 182 Amir Pnueli. The temporal logic of programs. In *FOCS*, pages 46–57, 1977. doi:10.1109/SFCS.1977.32.
- 183 Ameya Prabhu, Philip H. S. Torr, and Puneet K. Dokania. GDumb: A Simple Approach that Questions Our Progress in Continual Learning. In *ECCV (2)*, volume 12347 of *Lecture Notes in Computer Science*, pages 524–540. Springer, 2020. doi:10.1007/978-3-030-58536-5_31.
- 184 Luc De Raedt, Angelika Kimmig, and Hannu Toivonen. Problog: A probabilistic prolog and its application in link discovery. In Manuela M. Veloso, editor, *IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, January 6–12, 2007*, pages 2462–2467, 2007. URL: <http://ijcai.org/Proceedings/07/Papers/396.pdf>.
- 185 Kristo Raun, Riccardo Tommasini, and Ahmed Awad. I will survive: An event-driven conformance checking approach over process streams. In Valerio Schiavoni, Marcelo Pasin, Bettina Kemme, and Etienne Rivière, editors, *Proceedings of the 17th ACM International Conference on Distributed and Event-based Systems, DEBS 2023, Neuchatel, Switzerland, June 27–30, 2023*, pages 49–60. ACM, 2023. doi:10.1145/3583678.3596887.
- 186 Jesse Read, Albert Bifet, Bernhard Pfahringer, and Geoff Holmes. Batch-incremental versus instance-incremental learning in dynamic and evolving data. In *IDA*, volume 7619 of *Lecture Notes in Computer Science*, pages 313–323. Springer, 2012. doi:10.1007/978-3-642-34156-4_29.
- 187 Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H. Lampert. iCaRL: Incremental Classifier and Representation Learning. In *CVPR*, pages 5533–5542. IEEE Computer Society, 2017. doi:10.1109/CVPR.2017.587.

- 188 Raymond Reiter. On closed world data bases. In Hervé Gallaire and Jack Minker, editors, *Logic and Data Bases*, pages 55–76. Plenum Press, New York, 1978. doi:10.1007/978-1-4684-3384-5_3.
- 189 Xiangnan Ren and Olivier Curé. Strider: A hybrid adaptive distributed rdf stream processing engine. In *The Semantic Web—ISWC 2017: 16th International Semantic Web Conference, Vienna, Austria, October 21–25, 2017, Proceedings, Part I 16*, pages 559–576. Springer, 2017. doi:10.1007/978-3-319-68288-4_33.
- 190 Márcio Moretto Ribeiro. *Belief Revision in Non-Classical Logics*. Springer Briefs in Computer Science. Springer, 2013. doi:10.1007/978-1-4471-4186-0.
- 191 Alessandro Ronca, Mark Kaminski, Bernardo Cuenca Grau, and Ian Horrocks. The delay and window size problems in rule-based stream reasoning. *Artif. Intell.*, 306:103668, 2022. doi:10.1016/J.ARTINT.2022.103668.
- 192 Hans Rott. *Change, choice and inference: A study of belief revision and nonmonotonic reasoning*, volume 42 of *Oxford logic guides*. Oxford University Press, 2001.
- 193 Andrei A. Rusu, Neil C. Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive Neural Networks. *CoRR*, abs/1606.04671, 2016. doi:10.48550/arXiv.1606.04671.
- 194 Georgios M. Santipantakis, Akrivi Vlachou, Christos Doukeridis, Alexander Artikis, Ioannis Kontopoulos, and George A. Vouros. A Stream Reasoning System for Maritime Monitoring. In Natasha Alechina, Kjetil Nørnvåg, and Wojciech Penczek, editors, *25th International Symposium on Temporal Representation and Reasoning (TIME 2018)*, volume 120 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 20:1–20:17. Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.TIME.2018.20.
- 195 Matthias J. Sax, Guozhang Wang, Matthias Weidlich, and Johann-Christoph Freytag. Streams and tables: Two sides of the same coin. In Malú Castellanos, Panos K. Chrysanthis, Badrish Chandramouli, and Shimin Chen, editors, *Proceedings of the International Workshop on Real-Time Business Intelligence and Analytics, BIRTE 2018, Rio de Janeiro, Brazil, August 27, 2018*, pages 1:1–1:10. ACM, 2018. doi:10.1145/3242153.3242155.
- 196 Thomas Scharrenbach, Jacopo Urbani, Alessandro Margara, Emanuele Della Valle, and Abraham Bernstein. Seven commandments for benchmarking semantic flow processing systems. In Philipp Cimiano, Óscar Corcho, Valentina Presutti, Laura Hollink, and Sebastian Rudolph, editors, *The Semantic Web: Semantics and Big Data, 10th International Conference, ESWC 2013, Montpellier, France, May 26-30, 2013. Proceedings*, volume 7882 of *Lecture Notes in Computer Science*, pages 305–319. Springer, 2013. doi:10.1007/978-3-642-38288-8_21.
- 197 Patrik Schneider, Daniel Alvarez-Coello, Anh Le-Tuan, Manh Nguyen Duc, and Danh Le Phuoc. Stream reasoning playground. In Paul Groth, Maria-Esther Vidal, Fabian M. Suchanek, Pedro A. Szekely, Pavan Kapanipathi, Catia Pesquita, Hala Skaf-Molli, and Minna Tamper, editors, *The Semantic Web - 19th International Conference, ESWC 2022, Hersonissos, Crete, Greece, May 29 - June 2, 2022, Proceedings*, volume 13261 of *Lecture Notes in Computer Science*, pages 406–424. Springer, 2022. doi:10.1007/978-3-031-06981-9_24.
- 198 Jonathan Schwarz, Wojciech Czarnecki, Jelena Luketina, Agnieszka Grabska-Barwinska, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell. Progress & Compress: A scalable framework for continual learning. In *ICML*, volume 80 of *Proceedings of Machine Learning Research*, pages 4535–4544. PMLR, 2018. URL: <http://proceedings.mlr.press/v80/schwarz18a.html>.
- 199 Mario Scrocca, Riccardo Tommasini, Alessandro Margara, Emanuele Della Valle, and Sherif Sakr. The kaiju project: enabling event-driven observability. In Julien Gascon-Samson, Kaiwen Zhang, Khuzaima Daudjee, and Bettina Kemme, editors, *14th ACM International Conference on Distributed and Event-based Systems, DEBS 2020, Montreal, Quebec, Canada, July 13-17, 2020*, pages 85–96. ACM, 2020. doi:10.1145/3401025.3401740.
- 200 Joan Serrà, Didac Suris, Marius Miron, and Alexandros Karatzoglou. Overcoming Catastrophic Forgetting with Hard Attention to the Task. In *ICML*, volume 80 of *Proceedings of Machine Learning Research*, pages 4555–4564. PMLR, 2018. URL: <http://proceedings.mlr.press/v80/serra18a.html>.
- 201 Qi She, Fan Feng, Xinyue Hao, Qihan Yang, Chuanlin Lan, Vincenzo Lomonaco, Xuesong Shi, Zhengwei Wang, Yao Guo, Yimin Zhang, Fei Qiao, and Rosa H M Chan. OpenLORIS-Object: A Robotic Vision Dataset and Benchmark for Lifelong Deep Learning. *arXiv*, pages 1–8, 2019-11. doi:10.48550/arXiv.1911.06487.
- 202 Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual Learning with Deep Generative Replay. In *NIPS*, pages 2990–2999, 2017. doi:10.5555/3294996.3295059.
- 203 Utkarsh Srivastava and Jennifer Widom. Flexible time management in data stream systems. In *PODS*, pages 263–274. ACM, 2004. doi:10.1145/1055558.1055596.
- 204 Heiner Stuckenschmidt, Stefano Ceri, Emanuele Della Valle, and Frank Van Harmelen. Towards expressive stream reasoning. In *Dagstuhl Seminar Proceedings*. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2010. doi:10.4230/DagSemProc.10042.4.
- 205 Jakob Suchan, Mehul Bhatt, and Srikrishna Varadarajan. Out of sight but not out of mind: An answer set programming based online abduction framework for visual sensemaking in autonomous driving. In Sarit Kraus, editor, *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pages 1879–1885. ijcai.org, 2019. doi:10.24963/IJCAI.2019/260.
- 206 Jakob Suchan, Mehul Bhatt, and Srikrishna Varadarajan. Commonsense visual sensemaking

- for autonomous driving - on generalised neurosymbolic online abduction integrating vision and semantics. *Artif. Intell.*, 299:103522, 2021. doi:10.1016/j.artint.2021.103522.
- 207 Dan Suci, Dan Olteanu, Christopher Ré, and Christoph Koch. *Probabilistic databases*. Springer Nature, 2022. doi:10.1007/978-3-031-01879-4.
- 208 Gábor Szárnyas, Brad Bebee, Altan Birlir, Alin Deutsch, George Fletcher, Henry A. Gabb, Denise Gosnell, Alastair Green, Zhihui Guo, Keith W. Hare, Jan Hidders, Alexandru Iosup, Atanas Kiryakov, Tomas Kovatchev, Xinsheng Li, Leonid Libkin, Heng Lin, Xiaojian Luo, Arnau Prat-Pérez, David Püroja, Shipeng Qi, Oskar van Rest, Benjamin A. Steer, Dávid Szakállas, Bing Tong, Jack Waudby, Mingxi Wu, Bin Yang, Wenyuan Yu, Chen Zhang, Jason Zhang, Yan Zhou, and Peter Boncz. The linked data benchmark council (ldbc): Driving competition and collaboration in the graph data management space. In *TPCTC*, 2023. doi:10.48550/arXiv.2307.04350.
- 209 Gözde Ayşe Tataroğlu Özbek. Decentralized stream reasoning agents. In *Proceedings of the 17th ACM International Conference on Distributed and Event-based Systems*, pages 203–206, 2023. doi:10.1145/3583678.3603286.
- 210 Mahbod Tavallae, Ebrahim Bagheri, Wei Lu, and Ali A. Ghorbani. A detailed analysis of the KDD CUP 99 data set. In *CISDA*, pages 1–6. IEEE, 2009. doi:10.1109/CISDA.2009.5356528.
- 211 Douglas B. Terry, David Goldberg, David A. Nichols, and Brian M. Oki. Continuous queries over append-only databases. In Michael Stonebraker, editor, *Proceedings of the 1992 ACM SIGMOD International Conference on Management of Data, San Diego, California, USA, June 2-5, 1992*, pages 321–330. ACM Press, 1992. doi:10.1145/130283.130333.
- 212 Artem Thofimov, Igor E. Kuralenok, Nikiga Marshalkin, and Boris Novikov. Delivery, consistency, and determinism: rethinking guarantees in distributed stream processing. *CoRR*, abs/1907.06250, 2019. doi:10.48550/arXiv.1907.06250.
- 213 Edward Thomas, Jeff Z. Pan, and Yuan Ren. Trowl: Tractable OWL 2 reasoning infrastructure. In Lora Aroyo, Grigoris Antoniou, Eero Hyvönen, Annette ten Teije, Heiner Stuckenschmidt, Liliانا Cabral, and Tania Tudorache, editors, *The Semantic Web: Research and Applications, 7th Extended Semantic Web Conference, ESWC 2010, Heraklion, Crete, Greece, May 30 - June 3, 2010, Proceedings, Part II*, volume 6089 of *Lecture Notes in Computer Science*, pages 431–435. Springer, 2010. doi:10.1007/978-3-642-13489-0_38.
- 214 Mattias Tiger and Fredrik Heintz. Stream reasoning using temporal logic and predictive probabilistic state models. In *TIME*, pages 196–205, 2016. doi:10.1109/TIME.2016.28.
- 215 Mattias Tiger and Fredrik Heintz. Incremental reasoning in probabilistic signal temporal logic. *Int. J. Approx. Reason.*, 119:325–352, 2020. doi:10.1016/j.ijar.2020.01.009.
- 216 Riccardo Tommasini, Pieter Bonte, Femke Ongena, and Emanuele Della Valle. Rsp4j: an api for rdf stream processing. In *The Semantic Web: 18th International Conference, ESWC 2021, Virtual Event, June 6–10, 2021, Proceedings 18*, pages 565–581. Springer, 2021. doi:10.1007/978-3-030-77385-4_34.
- 217 Riccardo Tommasini, Pieter Bonte, Fabiano Spiga, and Emanuele Della Valle. *Streaming Linked Data: From Vision to Practice*. Springer Nature, 2023. doi:10.1007/978-3-031-15371-6.
- 218 Riccardo Tommasini, Davide Calvaresi, and Jean-Paul Calbimonte. Stream reasoning agents: Blue sky ideas track. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, pages 1664–1680, 2019. doi:10.5555/3306127.3331894.
- 219 Riccardo Tommasini, Emanuele Della Valle, Andrea Mauri, and Marco Brambilla. Rsplab: Rdf stream processing benchmarking made easy. In *The Semantic Web—ISWC 2017: 16th International Semantic Web Conference, Vienna, Austria, October 21-25, 2017, Proceedings, Part II 16*, pages 202–209. Springer, 2017. doi:10.1007/978-3-319-68204-4_21.
- 220 Transaction Processing Performance Council (TPC). *TPC-H Benchmark Specification*, 2023. URL: <http://www.tpc.org/tpch/>.
- 221 Georgia Troullinou, Haridimos Kondylakis, Matteo Lissandrini, and Davide Mottin. SOFOS: Demonstrating the Challenges of Materialized View Selection on Knowledge Graphs. In *Proceedings of the 2021 International Conference on Management of Data, SIGMOD/PODS '21*, pages 2789–2793, New York, NY, USA, 2021. Association for Computing Machinery. event-place: Virtual Event, China. doi:10.1145/3448016.3452765.
- 222 Efthymia Tsamoura, David Carral, Enrico Malizia, and Jacopo Urbani. Materializing Knowledge Bases via Trigger Graphs. *Proc. VLDB Endow.*, 14(6):943–956, 2021. doi:10.14778/3447689.3447699.
- 223 Jacopo Urbani, Markus Krötzsch, and Thomas Eiter. Chasing streams with existential rules. In Gabriele Kern-Isberner, Gerhild Lakemeyer, and Thomas Meyer, editors, *Proceedings of the 19th International Conference on Principles of Knowledge Representation and Reasoning (KR 2022)*, pages 416–421. IJCAI, 2022. URL: <https://proceedings.kr.org/2022/43/>.
- 224 Nithya N Vijayakumar and Beth Plale. Towards low overhead provenance tracking in near real-time stream filtering. In *International provenance and annotation workshop*, pages 46–54. Springer, 2006. doi:10.1007/11890850_6.
- 225 Raphael Volz, Steffen Staab, and Boris Motik. Incrementally Maintaining Materializations of Ontologies Stored in Logic Databases. *J. Data Semant.*, 2:1–34, 2005. doi:10.1007/978-3-540-30567-5_1.
- 226 Przemysław A. Walega, Mark Kaminski, and Bernardo Cuenca Grau. Reasoning over streaming data in metric temporal datalog. In *AAAI*, pages 3092–3099, 2019. doi:10.1609/aaai.v33i01.33013092.
- 227 Przemysław A Walega, Mark Kaminski, Dingmin Wang, and Bernardo Cuenca Grau. Stream reasoning with DatalogMTL. *Journal of Web Semantics*, 76:100776, 2023. doi:10.1016/j.websem.2023.100776.

- 228 Przemyslaw Andrzej Walega, David J. Tena Cuccala, Bernardo Cuenca Grau, and Egor V. Kostylev. The stable model semantics of datalog with metric temporal operators. *Theory and Practice of Logic Programming*, pages 1–35, 2023. doi:10.1017/S1471068423000315.
- 229 Przemyslaw Andrzej Walega, David J. Tena Cuccala, Egor V. Kostylev, and Bernardo Cuenca Grau. Datalogmtl with negation under stable models semantics. In Meghyn Bienvenu, Gerhard Lakemeyer, and Esra Erdem, editors, *Proceedings of the 18th International Conference on Principles of Knowledge Representation and Reasoning, KR 2021, Online event, November 3-12, 2021*, pages 609–618, 2021. doi:10.24963/kr.2021/58.
- 230 Dingmin Wang, Pan Hu, Przemyslaw Andrzej Walega, and Bernardo Cuenca Grau. MeTeoR: Practical reasoning in datalog with metric temporal operators. In *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, Thirty-Fourth Conference on Innovative Applications of Artificial Intelligence, IAAI 2022, The Twelfth Symposium on Educational Advances in Artificial Intelligence, EAAI 2022 Virtual Event, February 22 - March 1, 2022*, pages 5906–5913. AAAI Press, 2022. doi:10.1609/AAAI.V36I5.20535.
- 231 Guozhang Wang, Lei Chen, Ayusman Dikshit, Jason Gustafson, Boyang Chen, Matthias J. Sax, John Roesler, Sophie Blee-Goldman, Bruno Cadonna, Apurva Mehta, Varun Madan, and Jun Rao. Consistency and completeness: Rethinking distributed stream processing in apache kafka. In Guoliang Li, Zhanhuai Li, Stratos Idreos, and Divesh Srivastava, editors, *SIGMOD '21: International Conference on Management of Data, Virtual Event, China, June 20-25, 2021*, pages 2602–2613. ACM, 2021. doi:10.1145/3448016.3457556.
- 232 Min Wang, Marion Blount, John Davis, Archan Misra, and Daby Sow. A time-and-value centric provenance model and architecture for medical event streams. In *Proceedings of the 1st ACM SIGMOBILE international workshop on Systems and networking support for healthcare and assisted living environments*, pages 95–100, 2007. doi:10.1145/1248054.1248082.
- 233 Yi Wang and Joohyung Lee. Handling uncertainty in answer set programming. In Blai Bonet and Sven Koenig, editors, *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA*, pages 4218–4219. AAAI Press, 2015. doi:10.1609/aaai.v29i1.9726.
- 234 Zhun Yang, Adam Ishay, and Joohyung Lee. Neurasp: Embracing neural networks into answer set programming. In Christian Bessiere, editor, *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, pages 1755–1762. ijcai.org, 2020. doi:10.24963/ijcai.2020/243.
- 235 Huaxiu Yao, Caroline Choi, Bochuan Cao, Yoonho Lee, Pang Wei Koh, and Chelsea Finn. Wildtime: A benchmark of in-the-wild distribution shift over time. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022. URL: http://papers.nips.cc/paper_files/paper/2022/hash/43119db5d59f07cc08fca7ba6820179a-Abstract-Datasets_and_Benchmarks.html.
- 236 Carlo Zaniolo. Logical foundations of continuous query languages for data streams. In *Datalog*, pages 177–189, 2012. doi:10.1007/978-3-642-32925-8_18.
- 237 Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual Learning Through Synaptic Intelligence. In *International Conference on Machine Learning*, pages 3987–3995, 2017-07-17. doi:10.5555/3305890.3306093.
- 238 Shuhao Zhang, Juan Soto, and Volker Markl. A survey on transactional stream processing. *CoRR*, abs/2208.09827, 2022. doi:10.48550/arXiv.2208.09827.
- 239 Ying Zhang, Pham Minh Duc, Oscar Corcho, and Jean-Paul Calbimonte. Srbench: a streaming rdf/sparql benchmark. In *The Semantic Web—ISWC 2012: 11th International Semantic Web Conference, Boston, MA, USA, November 11-15, 2012, Proceedings, Part I 11*, pages 641–657. Springer, 2012. doi:10.1007/978-3-642-35176-1_40.
- 240 Giacomo Ziffer, Alessio Bernardo, Emanuele Della Valle, and Albert Bifet. Kalman filtering for learning with evolving data streams. In Yixin Chen, Heiko Ludwig, Yicheng Tu, Usama M. Fayyad, Xingquan Zhu, Xiaohua Hu, Suren Byna, Xiong Liu, Jianping Zhang, Shirui Pan, Vagelis Papalexakis, Jianwu Wang, Alfredo Cuzzocrea, and Carlos Ordonez, editors, *2021 IEEE International Conference on Big Data (Big Data)*, Orlando, FL, USA, December 15-18, 2021, pages 5337–5346. IEEE, 2021. doi:10.1109/BIGDATA52589.2021.9671365.
- 241 Giacomo Ziffer, Alessio Bernardo, Emanuele Della Valle, Vitor Cerqueira, and Albert Bifet. Towards time-evolving analytics: Online learning for time-dependent evolving data streams. *Data Science*, 6(1-2):1–16, 2022.
- 242 Indre Zliobaite, Albert Bifet, Jesse Read, Bernhard Pfahringer, and Geoff Holmes. Evaluation methods and decision theory for classification of streaming data with temporal dependence. *Mach. Learn.*, 98(3):455–482, 2015. doi:10.1007/s10994-014-5441-4.

Semantic Web: Past, Present, and Future

Ansgar Scherp   

Ulm University, Germany

Gerd Groener   

Carl Zeiss SMT GmbH, Oberkochen, Germany

Petr Škoda  

Department of Software Engineering, Faculty of Mathematics and Physics, Charles University, Prague, Czech Republic

Katja Hose   

TU Wien, Austria

Maria-Esther Vidal  

Leibniz University of Hannover, Germany

TIB-Leibniz Information Centre for Science and Technology, Hannover, Germany

Abstract

Ever since the vision was formulated, the Semantic Web has inspired many generations of innovations. Semantic technologies have been used to share vast amounts of information on the Web, enhance them with semantics to give them meaning, and enable inference and reasoning on them. Throughout the years, semantic technologies, and in particular knowledge graphs, have been used in search engines, data integration, enterprise settings, and machine learning.

In this paper, we recap the classical concepts and foundations of the Semantic Web as well as modern and recent concepts and applications, building upon these foundations. The classical topics

we cover include knowledge representation, creating and validating knowledge on the Web, reasoning and linking, and distributed querying. We enhance this classical view of the so-called “Semantic Web Layer Cake” with an update of recent concepts that include provenance, security and trust, as well as a discussion of practical impacts from industry-led contributions. We conclude with an outlook on the future directions of the Semantic Web.

This is a living document. If you like to contribute, please contact the first author and visit: <https://github.com/ascherp/semantic-web-primer>

2012 ACM Subject Classification Information systems → Semantic web description languages; Information systems → Markup languages; Computing methodologies → Ontology engineering; Computing methodologies → Knowledge representation and reasoning

Keywords and phrases Linked Open Data, Semantic Web Graphs, Knowledge Graphs

Digital Object Identifier 10.4230/TGDK.2.1.3

Category Survey

Related Version Lifelong version

Full Version: <https://github.com/ascherp/semantic-web-primer>

Funding *Maria-Esther Vidal*: Partially funded by Leibniz Association, program “Leibniz Best Minds: Programme for Women Professors”, project TrustKG-Transforming Data in Trustable Insights; Grant P99/2020.

Acknowledgements This article is a living document. The first version was written by Jannik, Scherp, and Staab in 2011 [88]. It was translated to German and updated by Gröner, Scherp, and Staab in 2013 [67] and updated again by Gröner and Scherp in 2020 [130]. This release in 2024 reflects on the latest developments in knowledge graphs and large language models.

Received 2023-07-03 **Accepted** 2024-01-29 **Published** 2024-05-03

Editors Aidan Hogan, Ian Horrocks, Andreas Hotho, and Lalana Kagal

Special Issue Trends in Graph Data and Knowledge – Part 2



© Ansgar Scherp, Gerd Groener, Petr Škoda, Katja Hose, and Maria-Esther Vidal; licensed under Creative Commons License CC-BY 4.0

Transactions on Graph Data and Knowledge, Vol. 2, Issue 1, Article No. 3, pp. 3:1–3:37



Transactions on Graph Data and Knowledge

TGDk Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

The vision of the Semantic Web as coined by Tim Berners-Lee, James Hendler, and Orla Lassila [19] in 2001 is to develop intelligent agents that can automatically gather semantic information from distributed sources accessible over the Web, integrate that knowledge, use automated reasoning [64], and solve complex tasks as such as schedule appointments in negotiation of the preferences of the involved parties. We have come a long way since then. In this paper, we reflect on the *past*, i. e., the ideas and components developed in the early days of the Semantic Web. Since the beginning, the Semantic Web has tremendously developed and undergone multiple waves of innovation. The Linked Data movement has especially seen uptake by industries, governments, and non-profit organizations, alike. We discuss those *present* components and concepts that have been added over the years and shown to be very useful. Although many concepts of the initial idea of the Semantic Web have been implemented and put into practice, still further research is needed to reach the full vision. Thus, this paper concludes with an outlook to *future* directions and steps that may be taken.

For the novice reader of the Semantic Web, we provide a brief historical overview of the developments and innovation waves of the Semantic Web: At the beginning of the Semantic Web, we were mainly talking about publishing Linked Data on the Web [73], i. e., semantic data typically structured using the Resource Description Framework (RDF)¹ that is accessible on the Web using URIs/IRIs to identify entities, classes, predicates, etc. By referencing entities from other websites and Web-accessible sources, i. e., dereferencable via HTTP, the data becomes naturally linked. By using standardized vocabularies and ontologies the information then becomes more aligned and easier to use across sources. These principles have allowed non-profit organizations, companies, governments, and individuals to publish and share large amounts of interlinked data, which has led to the success of the Linked Open Data cloud² since 2007. Since many of the large interconnected semantic sources are accessible via interfaces understanding structured query languages (SPARQL endpoints), federated query processing methods were developed that allow exploiting the strengths of structured query languages to precisely formulate an information need and optimize the query for efficient execution in a distributed setting.

When Google launched its Knowledge Graph in 2012³, semantic technologies experienced another wave of new applications in the context of searching information. Whereas search engines before mainly relied on keyword search and string-based matches of the keywords in the websites' text, the knowledge graph enabled including semantics to capture the user's information need as well as the meaning of potentially relevant documents. To achieve this purpose, Google's knowledge graph integrates large amounts of machine-processable data available on the Web and uses this information not only to improve search results but also to display infoboxes for entities identified in the user's keywords. It is only since 2012 that we have widely used the term "knowledge graph" to refer to semantic data, where entities are connected via relationships and form large graphs of interconnected information, typically with RDF as a common standard language. In recent years though (labeled) property graphs (LPG) have been used to manage knowledge graphs. We refer to the literature for a detailed comparison of RDF graphs and LPGs [78] and also like to point out that they can be converted into each other [23]. In this article, we consider knowledge graphs from the perspective of the Semantic Web, i. e., we consider RDF graphs.

¹ <http://www.w3.org/TR/rdf-primer/>

² <https://lod-cloud.net/>

³ <https://blog.google/products/search/introducing-knowledge-graph-things-not/>

A few years later, semantic technologies found another novel application in enterprise settings as enterprise knowledge graphs [55, 108] and data integration [33, 160]. Since all kinds of information can be structured as a graph, knowledge graphs can be used as a common structure to integrate heterogeneous information that is otherwise locked up in silos in different branches of a company. Integrating the data into a knowledge graph then allows for an integrated view, efficiently retrieving relevant information from this view once needed, and integrating external information that is available in the form of knowledge graphs, for instance on the Semantic Web. After all, online analytical processing-style queries can also be formulated with SPARQL⁴ and evaluated on (distributed) knowledge graphs.

In the past few years, learning on graph data became one of the fastest growing and most active areas in machine learning [71]. Graph representation learning has created a new wave of graph embedding models and graph neural networks on knowledge graphs for tasks such as entity classification and link prediction. Natural Language Processing (NLP) has been another important field of the Semantic Web since the early years to extract knowledge from textual data and make it machine readable. Another field where NLP meets the Semantic Web is user interfaces for search on structured data to enable intuitive, natural language querying for graph data [69] similar to web search engines. At the end of 2022, ChatGPT⁵ emerged as the first publicly available end-consumer tool based on a Large Language Model (LLM). Since then, the GPT-based family of LLMs has stirred up research and business alike and demonstrated impressive performance on many NLP tasks, including generating structured queries from user prompts and extracting structured knowledge from text [145].

The capabilities of tools like Bing Chat⁶ with underlying access to the World Wide Web are reminding one of the intelligent agents that were envisioned 20 years before. For example, at the time of writing, the GPT4-based tool Bing⁷ can internally generate SPARQL queries and execute them, while the structured response is seamlessly embedded into its natural language outputs to the users.⁸ While it already addresses some of the early visions of the Semantic Web, particularly the complex planning and reasoning capabilities of LLMs are – due to their nature of focusing on generating and processing text – still limited. We hypothesize that advances in neuro-symbolic AI and semantic technologies will be key for improving LLMs and bringing generative AI tools like Bing and the Semantic Web further together. We are keen to witness this next era of the Semantic Web.

In this paper, we provide a comprehensive overview of the Semantic Web with its semantic technologies and underlying principles that have been inspiring and driving the multiple waves of innovations in the past two decades. Section 2 provides a motivating example for the classic Semantic Web. We refer back to this example throughout the paper. Section 3 presents the principles and the general architecture of the Semantic Web along with the basic semantic technologies it is founded upon. Besides classical components, we are also describing recent developments, and pointing out components that are still being researched and developed. Section 4 shows how to represent distributed knowledge on the Semantic Web. The creation and maintenance of graph data is described in Section 5. Section 6 discusses the principle of reasoning and logical inference. Section 7 then shows how to query over the (distributed) graph data on the Semantic

⁴ <http://www.w3.org/TR/sparql11-query/>

⁵ <https://chat.openai.com/>

⁶ <http://bing.com>

⁷ <https://www.bing.com/>

⁸ Based on a sequence of prompts ran on January 22, 2024, using the GPT-4 model provided on the Bing Chat mobile app. The prompt sequence is: “do you have access to DBpedia”, “how do you access DBpedia”, “please give me an example where you access DBpedia in response”.

Web. We discuss the trustworthiness and provenance of data on the Semantic Web in Section 8. We provide extensive examples of applications based on and using the Semantic Web and its technologies in Section 9. Finally, we reflect on the impact the Semantic Web has on practitioners in Section 10. Finally, we conclude with a brief outlook on future developments for the Semantic Web.

2 Motivating Example

On the Semantic Web, knowledge components from different sources can be intelligently integrated with each other. As a result, complex questions can be answered, questions like “What types of music are played on British radio stations? At which time and day of the week?” or “Which radio station plays songs by Swedish artists?” In this section, we provide an overview of how the Semantic Web can be employed to answer those questions. We provide details of the components of the Semantic Web in the following sections.

We consider the example of the BBC program ontology with links to various other ontologies such as for music, events, and social networks as shown in Figure 1. We start with the BBC playlists of its radio stations. The playlists are published online in Semantic Web formats. We can leverage the playlist to get unique identifiers of played artists and bands. For example, the music group “ABBA” has a unique identifier in the form of a URI (<https://www.bbc.co.uk/programmes/b031yzpr>). This URI can be used to link the music group to information from the MusicBrainz⁹ music portal. MusicBrainz knows the members of the band, such as Benny Andersson, as well as the genre and songs. In addition, MusicBrainz is linked to Wikipedia¹⁰ (not shown in the figure), e.g., to provide information about artists, such as biographies on DBpedia [11]. Information about British radio stations can be found in the form of lists on Web pages such as Radio UK¹¹, which can also be converted into a representation in the Semantic Web.

We can see that the required information is distributed across multiple knowledge components, e.g., BBC Program, MusicBrainz, and others. Each knowledge component can in principle provide different access to the data and utilize various ways to describe the data. Consequently, to answer the questions the data must be integrated. On the Semantic Web, data integration relies on ontologies describing data and the meaning of relations in data.

Colloquially, an ontology is a description of concepts and their relationships. Ontologies are used to formally represent knowledge on the Semantic Web.¹² For example, Dublin Core¹³ provides a metadata schema for describing common properties of objects, such as the creator of the information, type, date, title, usage rights, and so on. Figure 1 presents ontologies used to describe data in our example. For example, the Playcount ontology¹⁴ of the BBC is used to model which artist was played and how many times in the programs. Ontologies can be interconnected in the Semantic Web. For example, the MusicBrainz ontology is connected to the BBC ontology using the Playcount ontology. Different ontologies with varying degrees of formality and different relationships to each other are used by the BBC to describe their data (see also [122]).

⁹ <http://musicbrainz.org/>

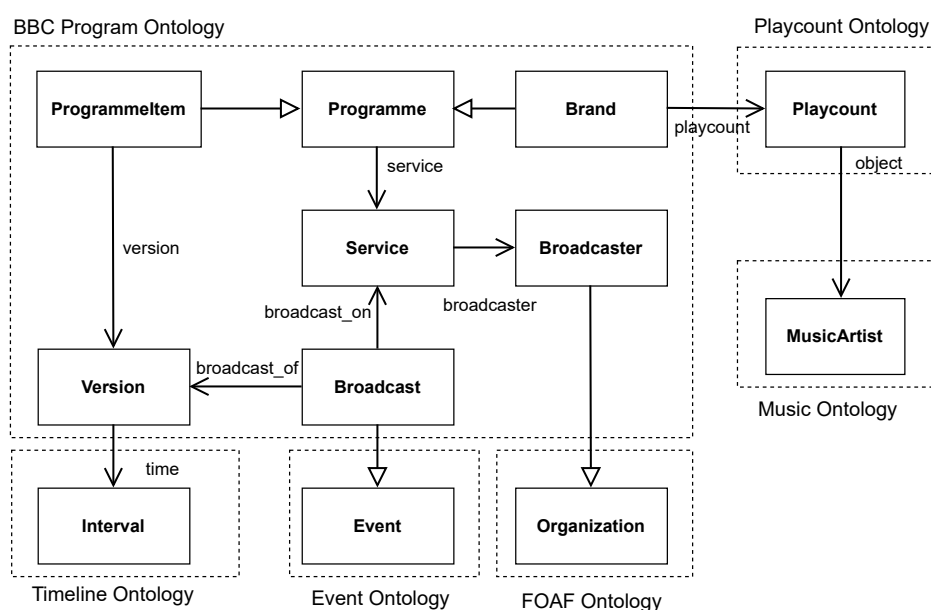
¹⁰ <https://www.wikipedia.org/>

¹¹ <https://www.radio-uk.co.uk>

¹² An ontology definition is provided in Section 4.2.

¹³ <http://dublincore.org/documents/dc-rdf/>

¹⁴ <http://dbtune.org/bbc/playcount/>



■ **Figure 1** Example of the BBC ontology with links to other ontologies (notation based on UML, here without prefix for namespaces).

As all this data is available and interconnected by ontologies, a user of the Semantic Web can directly ask for answers to questions in this and other scenarios. To make this possible, the Semantic Web requires generic software components, languages, and protocols that can interact seamlessly with each other. We introduce the classical and modern components of the architecture of the Semantic Web in Section 3.

In addition to the above example, the Semantic Web can be used for a variety of other applications (see examples in Section 9). Apart from technical aspects, the Semantic Web should also be understood as a socio-political phenomenon. Similar to the World Wide Web, various individuals and organizations publish their data on the Semantic Web and collaborate to link and improve this data. This impact on practitioners is discussed in Section 10.

3 Architecture of the Semantic Web

The example in Section 2 describes *what* the Semantic Web is as an infrastructure, but not *how* this is achieved. In fact, the capabilities of the Semantic Web *in a small scale* have already been implemented by some knowledge-based systems originating from artificial intelligence research, e.g., Heinsohn et al [76]. However, for the implementation of the vision *on a large scale*, i.e., the Web, these knowledge-based systems lacked flexibility, robustness, and scalability. In part, this was due to the complexity of the algorithms used. For example, knowledge bases in description logic in the 1990s, which serve as the basis of Web ontologies, were limited regarding their size such that they could handle at the most some hundred concepts [76].

In the meantime, enormous improvements have been achieved. Greatly increased computational power and optimized algorithms allow a practical handling of large ontologies like Simple Knowledge Organization System (SKOS)¹⁵, Gene Ontology¹⁶, Schema.org, and SNOMED-CT¹⁷. However,

¹⁵ <https://www.w3.org/TR/skos-reference/>

¹⁶ <https://geneontology.org/>

¹⁷ <https://www.snomed.org/>

there are some fundamental differences between traditional knowledge-based systems and the Semantic Web. Data management in traditional knowledge-based systems has weaknesses in terms of handling large amounts of data and data sources, among other things because of different underlying formalisms, distributed locations, different authorities, different data quality, and a high frequency of change in the data used.

The Semantic Web applies fundamental principles to deal with these problems; they represent the basis for the architecture of the Semantic Web. This architecture's building blocks can roughly be categorized into groups, covering the entire life cycle of handling and managing graph data on the Web. These groups are graph data representation, creation and validation of graph data, reasoning over and linking of graph data, (distributed) querying of graph data, crypto, provenance, and trustworthiness of graph data, and user interfaces and applications.

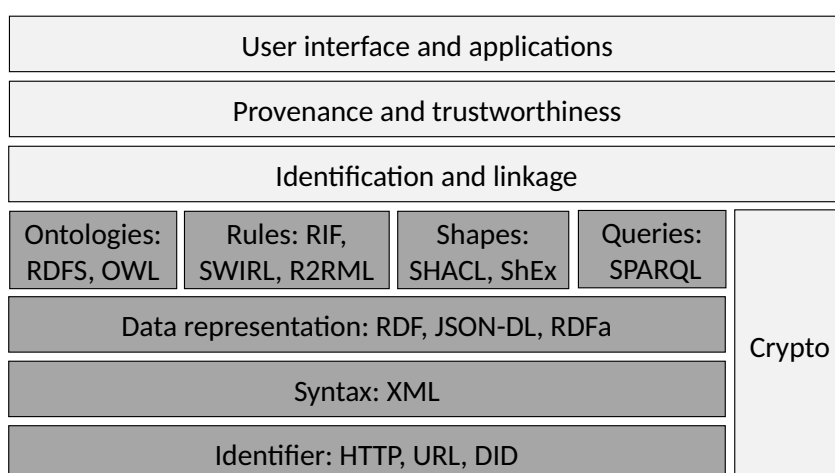
Below, we first introduce the principles of the Semantic Web, from which we derive the architecture and its main components. Subsequently, we describe the groups of the architecture. The principles of the Semantic Web are:

1. *Explicit and simple data representation*: A general data representation abstracts from the underlying formats and captures only the essentials.
2. *Distributed systems*: A distributed system operates on a large set of data sources without centralized control that regulates which information belongs where and to whom.
3. *Cross-references*: The advantages of a network of data in answering queries are not based solely on the sheer quantities of data but on their interconnection, which allows reusing data and data definitions from other sources.
4. *Loose coupling with common language constructs*: The World Wide Web and likewise the Semantic Web are mega-systems, i. e., systems consisting of many subsystems, which are themselves large and complex. In such a mega-system, individual components must be loosely coupled in order to achieve the greatest possible flexibility. Communication between the components is based on standardized protocols and languages, whereby these can be individually adapted to specific systems.
5. *Easy publishing and easy consumption*: Especially in a mega-system, participation, i. e., publishing and consumption of data, must be as simple as possible.

These principles are achieved through a mix of protocols, language definitions, and software components. Some of these components have already been standardized by the W3C, which has defined both syntax and formal semantics of languages and protocols. Other components are not yet standardized, but they are already provided for the so-called *Semantic Web Layer Cake* by Tim Berners-Lee (cf. <http://www.w3.org/2007/03/layerCake.png>). We present a variant of the Semantic Web architecture, distinguishing between standardized languages and current developments. A graphical representation of the architecture can be found in Figure 2.

Identifier for Resources: HTTP, URL, DID

Entities (also called resources) are identified on the Internet by so-called Uniform Resource Identifiers (URIs) [17]. When a URI holds a dereferenceable location of the resource, in other words, it can be employed to get access to the resource via HTTP, it is called a Uniform Resource Locator (URL) [20, 18]. Furthermore, Internationalized Resource Identifiers (IRIs) [45] supplement URIs with international character sets from Unicode/ISO10646. URIs are globally and universally used but are usually not under our control. A recent W3C recommendation, the Decentralized Identifiers (DIDs), introduces an alternative approach to the above identifiers [137]. A DID is by default decentralized and allows for self-sovereign management of the identity, i. e., the control of a DID and its associated data is with the users.



■ **Figure 2** Representation of the components of the so-called “Semantic Web Layer Cake”. W3C language standards are shown in dark gray. Current developments are shown in light gray.

In our example in Section 2, a URI ¹⁸ describes the musician Benny Andersson of the Swedish pop group ABBA. A user can dereference a URI that refers to ABBA, e.g., by performing a so-called look-up using HTTP to obtain a detailed description of the URI. We refer to the referenced standards for details.

For a detailed discussion of the role of dereferenceable URIs on the Semantic Web, we refer to the Linked Data principles described in Section 4.1.

Syntax for Data Exchange: XML, JSON-LD, RDFa

The Extensible Markup Language (XML)¹⁹ is used to structure documents and enables the specification and serialization of structured data. In addition, other data formats were introduced to facilitate for serialization of RDF data, often replacing XML. We can view those formats as forming two groups. The first group consists of formats designed specifically for RDF data, such as Turtle²⁰, N-triple²¹, and TRIG²². These are easier to view in a text editor, compared to XML, and thus easier to understand and modify. While initially not included in standards, their popularity has led to them being official W3C recommendations since 2014. The other group of formats is built by extending existing data formats. As a result, those can be employed to add RDF to existing systems. Examples of such formats are JSON-LD²³, CSV on the Web (CSVW)²⁴, and RDFa²⁵ extending JSON, CSV, and (X)HTML, respectively.

Graph Data Representation: RDF

In addition to the referencing of resources and a uniform syntax for the exchange of data, a data model is required that allows resources to be described both individually and in their entirety

¹⁸ <http://www.bbc.co.uk/music/artists/2f031686-3f01-4f33-a4fc-fb3944532efa#artist>

¹⁹ <https://www.w3.org/TR/xml/>

²⁰ <https://www.w3.org/TR/turtle/>

²¹ <https://www.w3.org/TR/n-triples/>

²² <https://www.w3.org/TR/trig/>

²³ <https://www.w3.org/TR/json-ld/>

²⁴ <https://www.w3.org/TR/tabular-data-primer/>

²⁵ <https://www.w3.org/TR/rdfa-primer/>

and how they are linked [73, 75]. An integrated representation of data from multiple sources is provided by a data model based on directed graphs [114]. The corresponding W3C standard language is RDF (Resource Description Framework)²⁶.

An RDF graph consists of a set of RDF triples, where a triple consists of a subject, predicate (property), and object. An RDF graph can be given an identifier, such a graph is called a named graph [29]. RDF graphs can be serialized in several ways (see Syntax for Data Exchange above). It is important to note that some formats (e. g., Turtle) do not support named graphs. Finally, RDF-star (aka RDF*)^{27,28} was introduced to allow nesting of triples and thus enables an efficient way to make statements about statements while avoiding reification and the increased number of triples and complexity that come along with it.

The representation of graph data is further discussed in Section 4.

Creation and Validation of Graph Data: RIF, SWRL, [R2]RML, and SHACL

In the RDF context, a rule is a logical statement employed to infer new facts from existing graph data or to validate the data itself. RIF (Rule Interchange Format)²⁹ is a W3C recommendation format designed to facilitate the seamless interchange of rules between different rule engines. This enables the extraction of rules from one engine, their translation into RIF, publication, and subsequent conversion into the native syntax of another rule engine for execution. SWRL³⁰ is a rule-based language designed for representing complex relationships and reasoning.

Rules can also be used to state the correspondence between data sources and RDF graphs. The RDB to RDF Mapping Language (R2RML)³¹ and the RDF Mapping Language (RML)³² correspond to rule-based mapping languages for the declarative definition of RDF graphs. R2RML is the W3C recommendation for representing mappings from relational databases to RDF datasets, while RML extends R2RML to express rules not only from relational databases but also from data in the format of CSV, JSON, or XML.

Validating constraints, representing syntactic and semantic restrictions in RDF graphs, is essential for ensuring data quality. In addition to rule-based languages, shapes allow for the specification of conditions to meet data quality criteria and integrity constraints. A shape encompasses a conjunction of constraints representing conditions that nodes in an RDF graph must satisfy [79]. A shapes graph is a labeled directed graph where nodes correspond to shapes, and edges denote interrelated constraints. The Shapes Constraint Language (SHACL) [95] and Shape Expressions (ShEx) [117]) are two W3C-recommendations to express shapes graphs over RDF [119].

The creation and validation of graph data are described in detail in Section 5.

Reasoning and Linking of Graph Data: RDFS, OWL

Data from different sources may be heterogeneous. In order to deal with this heterogeneity and to model the semantic relationships between resources, the RDF Schema (RDFS)³³ vocabulary

²⁶<https://www.w3.org/RDF/>

²⁷<https://w3c.github.io/rdf-star/>

²⁸<https://blog.liu.se/olafhartig/2019/01/10/position-statement-rdf-star-and-sparql-star/>

²⁹http://www.w3.org/2005/rules/wiki/RIF_Working_Group

³⁰<https://www.w3.org/submissions/SWRL/>

³¹<https://www.w3.org/TR/r2rml/>

³²<https://rml.io/specs/rml/>

³³<https://www.w3.org/TR/rdf11-schema/>

extends RDF by modeling types of resources (so-called RDF classes) and semantic relationships on types and properties in the form of generalizations and specializations. Likewise, it can be used to model the domain and range of properties.

RDFS is not expressive enough to merge data from different sources and define consistency criteria about it, such as the disjointness of classes or the equivalence of resources. The Web Ontology Language (OWL)³⁴ [155] is an ontology language with formally defined meaning based on description logic. This allows for reasoning services to be provided by knowledge-based systems for OWL ontologies. OWL can be exchanged using RDF data formats. Compared to RDFS, OWL provides more expressive language constructs. For example, OWL allows the specification of equivalences between classes and cardinality constraints on properties [155].

Reasoning over RDF graphs, incorporating RDFS and OWL models enhances semantic expressiveness and inferential capabilities. This involves making implicit information explicit, inferring new triples, and validating the RDF graph's consistency against defined ontological constraints. RDFS provides basic entailment regimens, creating hierarchies and simple inferencing via sub-class and sub-property relationships. In contrast, OWL introduces advanced constructs such as property characteristics (e. g., functional, inverse, symmetric properties), cardinalities, and disjointness axioms, enabling more expressive and complex modeling. Integrating RDFS and OWL reasoning mechanisms empowers applications to derive insights, discover implicit knowledge, and ensure adherence to specified ontological constraints within RDF-based knowledge representations.

Graph data aggregated from many data sources, such as in our example in Section 2, may contain many different identities. But those identities may represent the same set of real-world objects. Integration and linkage mechanisms allow references to be made between data from different sources. A popular approach to state the identity of two resources v and w is the `owl:sameAs` feature of OWL.

We discuss the reasoning over and linking of graph data in Section 6.

Querying of Graph Data: SPARQL

Since RDF makes it possible to integrate data from different sources, a query language is needed that allows formulating queries over individual RDF graphs as well as over the combination of multiple RDF graphs across multiple sources. SPARQL³⁵ (a recursive acronym for SPARQL Protocol and RDF Query Language) is a declarative query language for RDF graphs that enables us to formulate such queries. SPARQL 1.1³⁶ is the current version of SPARQL, which includes the capability to formulate federated queries over distributed data sources.

The basic building blocks of a SPARQL query are triple and graph patterns. A triple pattern corresponds to an RDF triple but where one, two, or all three of its components are replaced by variables (denoted with a leading "?"). These triple patterns with variables are to be matched in the queried graph. Multiple triple patterns can be combined into more complex graph patterns describing the connections between multiple nodes in the graph. The solution to such a SPARQL query then corresponds to all the subgraphs in an RDF graph matching this pattern.

Finally, there is RDF-star (aka RDF*)^{37,38} – along with the corresponding SPARQL-star/SPARQL* extension – was proposed and since then was implemented by several triple stores [2] that often provide publicly accessible SPARQL endpoints. The key idea with RDF/SPARQL-star

³⁴<https://www.w3.org/OWL/>

³⁵<http://www.w3.org/TR/rdf-sparql-query/>

³⁶<http://www.w3.org/TR/sparql11-query/>

³⁷<https://w3c.github.io/rdf-star/>

³⁸<https://blog.liu.se/olafhartig/2019/01/10/position-statement-rdf-star-and-sparql-star/>

3:10 Semantic Web: Past, Present, and Future

is to allow the nesting of triples to enable an efficient way to allow statements about statements while avoiding reification and the increased number of triples and complexity that come along with it.

We describe SPARQL and federated querying in Section 7.

Crypto, Provenance, and Trustworthiness of Graph Data

Other aspects of the Semantic Web are encryption and authentication to ensure that data transmissions cannot be intercepted, read, or modified. Crypto modules, such as SSL (Secure Socket Layer), verify digital certificates and enable data protection and authentication. In addition, there are digital signatures for graphs that integrate seamlessly into the architecture of the Semantic Web and are themselves modeled as graphs again [93]. This allows graph signatures to be applied iteratively and enables to building trust networks. The Verifiable Credentials Data Model, a recent W3C recommendation, introduces a standard to model trustworthy credentials for graphs on the web³⁹. Data on the Semantic Web can be augmented with additional information about its trustworthiness and provenance.

Aspects of trustworthiness and provenance of graph data as well as crypto are discussed in Section 8.

User Interfaces and Applications

A user interface enables users to interact with data on the Semantic Web. From a functional perspective, some user interfaces are generic and operate on the graph structure of the data, whereas others are tailored to specific tasks, applications, or ontologies. New paradigms are exploring the spectrum of possible user interfaces between generality and specific end-user requirements.

Semantic Web applications are discussed in Section 9. The impact on practitioners is described in Section 10.

4 Representation of Graph Data

The Linked Open Data principles are notably the most successful and widely adopted choice for representing RDF graph data on the web. Thus, we first introduce the reader to how to represent graph data as Linked Data. Subsequently, we introduce the notion of ontologies. This is followed by a more detailed analysis of the different types of ontologies. We give examples of ontologies throughout the sections. With this background in mind, we reconsider our running example from Section 2 and analyze the given distributed network of ontologies. In this context, we also introduce and discuss the notion of ontology design patterns.

4.1 Linked Graph Data on the Web

The *Linked Data* principles⁴⁰ define the methods for representing, publishing, and using data on the Semantic Web. They can be summarized as follows:

1. URIs are used as names for entities.
2. The HTTP protocol's GET method is used to retrieve descriptions for a URI.
3. Data providers shall return relevant information in response to HTTP GET requests on URIs using standards, e. g., in RDF.
4. Links to other URIs shall be used to facilitate knowledge discovery and use of additional information.

³⁹ <https://www.w3.org/TR/vc-data-model/>

⁴⁰ <http://www.w3.org/DesignIssues/LinkedData.html>

Publishing data using Linked Data principles allows easy access to data via HTTP. This allows exploration of resources and navigation across resources on the Semantic Web. URIs (see 1.) are dereferenced using HTTP requests (2.) to obtain additional information about a given resource. In particular, using standardized syntax (3.), this information may also contain links to other resources (4.).

Figure 3 represents an example of Linked Data about the pop group ABBA. The example describes several relationships linking entities to ABBA’s URI, such as `foaf:member` and `rdf:type`. In the figure “ABBA”, or more precisely the URI of ABBA, is the subject, “Property” refers to relationships, and “Value” represents objects of the RDF triples. The relation `owl:sameAs` will be explained in more in Section 6. The prefixes `foaf`, `rdf`, and `owl` refer to vocabularies of the FOAF ontology⁴¹, and the W3C language specifications of RDF and OWL, respectively.

ABBA	
Resource URI: http://dbtune.org/musicbrainz/resource/artist/d87e52c5-bb8d-4da8-b941-9f4928627dc8	
Property	Value
<code>vocab:alias</code>	Abba
<code>vocab:alias</code>	Ebjörn + Benny + Anna + Frieda
<code>bio:event</code>	< http://dbtune.org/musicbrainz/resource/artist/d87e52c5-bb8d-4da8-b941-9f4928627dc8/birth >
<code>bio:event</code>	< http://dbtune.org/musicbrainz/resource/artist/d87e52c5-bb8d-4da8-b941-9f4928627dc8/death >
<code>foaf:homepage</code>	< http://www.abbsite.com >
<code>foaf:member</code>	< http://dbtune.org/musicbrainz/resource/artist/042c35d3-0756-4804-b2c2-be57a683efa2 >
<code>foaf:member</code>	< http://dbtune.org/musicbrainz/resource/artist/2f031686-3f01-4f33-a4fc-fb3944532efa >
<code>foaf:member</code>	< http://dbtune.org/musicbrainz/resource/artist/aebbb417-0d18-4fec-a2e2-ce9663d1fa7e >
<code>foaf:member</code>	< http://dbtune.org/musicbrainz/resource/artist/ffb77292-9712-4d03-94aa-bdb1d4771d38 >
<code>mo:musicbrainz</code>	< http://musicbrainz.org/artist/d87e52c5-bb8d-4da8-b941-9f4928627dc8 >
<code>foaf:name</code>	ABBA
<code>owl:sameAs</code>	< http://dbpedia.org/resource/ABBA >
<code>owl:sameAs</code>	< http://sv.wikipedia.org/wiki/Abba >
<code>owl:sameAs</code>	< http://www.bbc.co.uk/music/artists/d87e52c5-bb8d-4da8-b941-9f4928627dc8#artist >
<code>rdf:type</code>	<code>mo:MusicArtist</code>

■ **Figure 3** Linked Data example for ABBA.

4.2 Ontologies

An ontology is commonly defined as a formal, machine-readable representation of key concepts and relationships within a specific domain [111, 109]. In essence, ontologies capture a shared perspective [111] that is, the formal conceptualization of ontologies expresses a consensus view among different stakeholders. Visualizing ontologies is akin to viewing a spectrum, with a specificity of concepts, their relationships, and the granularity of meaning representation varying along this continuum [101, 149, 148]. A controlled vocabulary corresponds to the less expressive form of ontology, comprising a restrictive list of words or terms used for labeling, indexing, or categorization. The Clinical Data Interchange Standards Consortium (CDISC) Terminology is an

⁴¹<http://xmlns.com/foaf/spec/>

exemplary vocabulary that harmonizes definitions in clinical research.⁴² A thesaurus is located next in the spectrum; they enhance controlled vocabularies with information about terms and their synonyms and broader/narrower relationships. The Unified Medical Language System (UMLS) integrates medical terms and their synonyms.⁴³ Next, taxonomies are built over controlled vocabularies to provide a hierarchical structure, e. g., parent/child relationship. SNOMED-CT⁴⁴ (Systematized Nomenclature of Medicine Clinical Terms) provides a terminology and coding system used in healthcare and medical fields; medical concepts organized in a hierarchical structure enabling a granular representation of clinical information. The Simple Knowledge Organization System (SKOS)⁴⁵ is a W3C standard to describe knowledge about organizational systems. Lastly, ontologies are at the highest extreme of the spectrum, integrating sets of concepts with attributes and relationships to define a domain of knowledge.

Note, SKOS is a popular standard for modeling domain-specific taxonomies in the different scientific communities such as economics, social sciences, etc. to represent concepts and their relationships, most importantly narrower, broader, and related. However, it does not have the expressiveness of OWL with its complex expressions on classes and relations. For a detailed discussion, we refer to the literature such as [89] and the W3C on using OWL and SKOS⁴⁶.

4.3 Types and Examples of Ontologies

A network of ontologies, such as the example shown in Figure 1, may consist of a variety of ontologies created by different actors and communities. Ontologies may be the result of a transformation or reengineering activity of a legacy system, such as a relational database or existing taxonomy such as the Dewey Decimal Classification⁴⁷ or Dublin Core. Other ontologies are created from scratch. This involves applying existing methods and tools for ontology engineering and choosing an appropriate representation language for the ontology (see Section 6).

Ontology engineering deals with the methods for creating ontologies [65] and has its origins in software engineering in the creation of domain models and in database design in the creation of conceptual models. A good overview of ontology engineering can be found in several reference books [65]. Ontologies vary greatly in their structure, size, development methods applied, and application domains considered. Complex ontologies are also distinguished in terms of their purpose and granularity.

Domain Ontologies represent knowledge specific to a particular domain [48, 109]. Domain ontologies are used as external sources of background knowledge [48]. They can be built on foundational ontologies [110] or core ontologies [131], which provide precise structuring to the domain ontology and thus improve interoperability between different domain ontologies. Domain ontologies can be simple such as the FOAF ontology or the event ontology mentioned above, or very complex and extensive, having been developed by domain experts, such as the SNOMED medical ontology.

Core Ontologies represent a precise definition of structured knowledge in a particular domain spanning multiple application domains [131, 109]. Examples of core ontologies include core ontologies for software components and web services [109], for events and event relationships [129], or for multimedia metadata [126]. Core ontologies should thereby build on foundational ontologies

⁴² <https://datascience.cancer.gov/resources/cancer-vocabulary/cdisc-terminology>

⁴³ <https://www.nlm.nih.gov/research/umls/index.html>

⁴⁴ <https://www.snomed.org/value-of-snomedct>

⁴⁵ <https://www.w3.org/TR/skos-reference/>

⁴⁶ <https://www.w3.org/2006/07/SWD/SKOS/skos-and-owl/master.html>

⁴⁷ <http://dewey.info/>

to benefit from their formalization and strong axiomatization [131]. For this purpose, new concepts and relations are added to core ontologies for the application domain under consideration and are specialized by foundational ontologies.

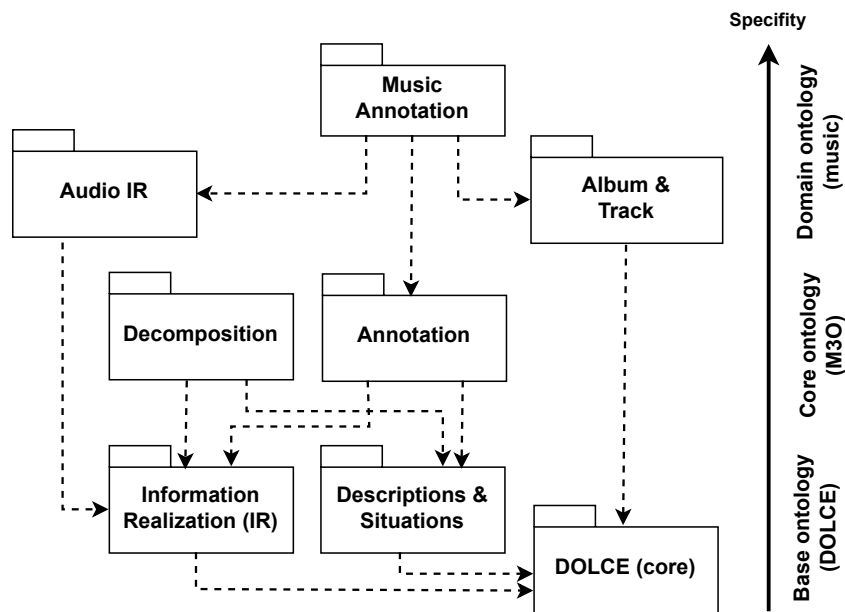
Foundational Ontologies have a very wide scope and can be reused in a wide variety of modeling scenarios [24]. They are therefore used for reference purposes [109] and aim to model the most general and generic concepts and relations that can be used to describe almost any aspect of our world [24, 109], such as objects and events. An example is the Descriptive Ontology for Linguistic and Cognitive Engineering (DOLCE) [24]. Such basic ontologies have a rich axiomatization that is important at the developmental stage of ontologies. They help ontology engineers to have a formal and internally consistent conceptualization of the world, which can be modeled and checked for consistency. For the use of foundational ontologies in a concrete application, i. e., during the runtime of an application, the rich axiomatization can often be removed and replaced by a more lightweight version of the foundational ontology.

In contrast, domain ontologies are built specifically to allow automatic reasoning at runtime. Therefore, when designing and developing ontologies, completeness and complexity on the one hand must always be balanced with the efficiency of reasoning mechanisms on the other. In order to represent structured knowledge, such as the scenario depicted in Figure 1, interconnected ontologies are needed, which are spanned in a network over the Internet. For this purpose, the ontologies used must match and be aligned with each other.

4.4 Distributed Network of Ontologies and Ontology Patterns

A network of ontologies must be flexible with respect to the functional requirements imposed on it. This is because systems are modified, extended, combined, or integrated over time. In addition, the networked ontologies must lead to a common understanding of the modeled domain. This common understanding can be achieved through a sufficient level of formalization and axiomatization, and through the use of ontology patterns. An ontology pattern, similar to a design pattern in software engineering, represents a generic solution to a recurring modeling problem [131]. Ontology patterns allow to select parts from the original ontology. Either all or only certain patterns of an ontology can be reused in the network. Thus, to create a network of ontologies, e. g., existing ontologies and ontology patterns can be merged on the Web. The ontology engineer can drive or explicitly provide for the modularization of ontologies using ontology patterns. Core ontologies represent one approach to designing a network of ontologies (see in detail [131]). They allow to capture and exchange structured knowledge in complex domains. Well-defined core ontologies fulfill the properties mentioned in the previous section and allow easy integration and smooth interaction of ontologies (see also [131]). The networked ontologies approach leads to a flat structure, as shown in Figure 1, where all ontologies are used on the same level. Such structures can be managed up to a certain level of complexity.

The approach of networked core ontologies is illustrated by the example of ontology layers starting from foundational to core to domain ontologies. As shown in Figure 4, DOLCE is the foundational ontology at the bottom layer, the Multimedia Metadata Ontology (M3O) [126] as the core ontology for multimedia metadata, and an extension of M3O for the music domain. Core ontologies are typically defined in description logic and cover a field larger than the specific application domain requires [57]. Concrete information systems will typically use only a subset of core ontologies. To achieve modularization of core ontologies, they should be designed using ontology patterns. By precisely matching the concepts in the core ontology with the concepts provided in the foundational ontology, they provide a solid foundation for future extensions. New patterns can be added and existing patterns can be extended by specializing the concepts and roles. Figure 4 shows different patterns of the M3O and DOLCE ontologies.



■ **Figure 4** Ontology layers combining the foundational ontology DOLCE, the multimedia metadata ontology M3O, domain-specific extensions to M3O for annotating audio data and music, and a domain ontology for albums and tracks.

Ideally, the ontology patterns of the core ontologies are reused in the domain ontologies [57], as shown in Figure 4. However, since it cannot be assumed that all domain ontologies are aligned with a foundational or core ontology, the option that domain ontologies are developed and maintained independently must also be considered. In this case, domain knowledge can be reused in core ontologies by applying the Descriptions and Situations (DnS) ontology pattern of the foundational ontology DOLCE. The DnS ontology pattern is an ontological formalization of context [109] by defining different views using roles. These roles can refer to domain ontologies and allow a clear separation of the structured knowledge of the core ontology and domain-specific knowledge. To model a network of ontologies, such as the example described above, the Web Ontology Language (OWL) and its ability to axiomatize using description logic [12] is used. In addition to being used to model a distributed knowledge representation and integration, OWL, is also used in particular to derive inferences from this knowledge, which is described in Section 6.

5 Creation and Validation of Graph Data

In this section, we describe the creation of graph data from legacy data. Many tools are available for this task, which support various mappings and transformations. Subsequently, we discuss data quality and the validation of knowledge graphs, including the recent approaches on shapes. We also reflect on the role of the open-world versus closed-world assumption with respect to validating data.

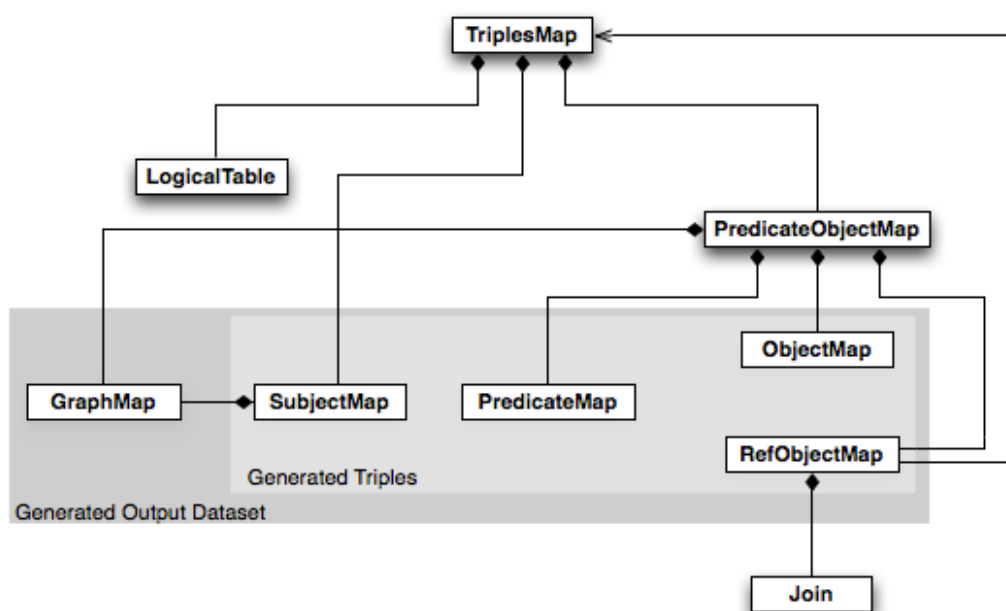
5.1 Graph Data Creation

Graph data can be created by transforming legacy data via a data integration system [98], which consists of a unified schema, data sources, and mapping rules. These mapping rules define the

concepts within the schema and establish links to the data sources. By employing declarative definitions, knowledge graph creation promotes modularity and reusability. This approach allows users to trace the entire graph creation process, leading to improved transparency and ease of maintenance.

To enable comprehensive and extensive graph specification, mappings and transformations have been developed to convert data from various storage models into Semantic Web data models like RDF. These mappings and transformations facilitate the mapping of data into RDF, thereby supporting the integration of diverse data sources into the Semantic Web.

The mapping language R2RML [37] defines mapping rules from relational databases (relational data models) to RDF graphs. These mappings themselves are also RDF triples [15]. Because of its compact representation, Turtle is considered a user-friendly notation of RDF graphs. The structure of R2RML is illustrated in Figure 5; essentially, table contents are mapped to triples by the classes `SubjectMap`, `PredicateMap`, and `ObjectMap`. If the object is a reference to another table, this reference is called `RefObjectMap`. Here, `SubjectMap` contains primary key attributes of the corresponding table. Thus, there exists a mapping rule representable in RDF graphs by means of which tables of relational databases can be represented as RDF graphs.



■ **Figure 5** Structure of a relational data mapping (source: [37]).

The RDF Mapping Language (RML)[42] extends R2RML to encompass the definition of logical sources in various formats, including CSV, JSON, XML, and HTML. This enhancement enables RML to introduce new operators that facilitate the integration of data from diverse sources into the Semantic Web. Thus, instead of `LogicalTable`, RML includes the tag `LogicalSource`, to allow for the retrieval of data in several formats. Additionally, RML resorts to W3C-standardized vocabularies and enables the definition of retrieval procedures to collect data from Web APIs or databases. R2RML and RDF mapping rules are expressed in RDF, and their graphs document how classes and properties in one or various ontologies that are part of an RDF graph are populated from data collected from potentially heterogeneous data sources.

Over time, the Semantic Web community has actively contributed to addressing the challenge of integrating heterogeneous datasets, resulting in the development of several frameworks for executing declarative mapping rules [34, 28, 116]. A rich spectrum of tools (e. g., RMLMapper [42], RocketRML [136], CARML⁴⁸, SDM-RDFizer [87], Morph-KGC [8], and RMLStreamer [112]) offers the possibility of executing R2RML and RML rules and efficiently materializing the transformed data into RDF graphs. Van Assche et al. [10] provided an extensive survey detailing the main characteristics of these engines. Despite significant efforts in developing these solutions, certain parameters can impact the performance of the graph creation process [31]. Existing engines may face challenges when handling complex mapping rules or large data sources. Nonetheless, the community continues to collaborate and address these issues. An example of such collaboration is the Knowledge Graph Construction Workshop 2023 Challenge⁴⁹ that took place at ESWC 2023. This community event aims to understand the strengths and weaknesses of existing approaches and devise effective methods to overcome existing limitations.

RDF graphs can also be dynamically created through the execution of queries over data sources. These queries involve the rewriting of queries expressed in terms of an ontology, based on mapping rules that establish correspondences between data sources and the ontology. Tools such as Ontop [28], Ultrawrap [135], Morph [116], Squerall [100], and Morph-CSV [32] exemplify systems that facilitate the virtual creation of RDF graphs.

5.2 Quality and Validation of Graph Data

Quality and validation of the graph data are crucial to maintaining the integrity of the Semantic Web [44, 38, 163]. The evaluation of integrity constraints allows for the identification of inconsistencies, inaccuracies, or contradictions within the data. They also help maintain consistency by ensuring related data elements remain coherent. Constraints are logical statements – expressed in a particular language – that impose restrictions on the values taken for target nodes in a given property.

Constraints can be expressed using OWL [144], SPARQL queries [97], or using shapes. However, the interpretation of the results depends on the semantics followed to interpret the failure of an integrity constraint. For example, constraints expressed in OWL are validated using an Open-World Assumption (OWA) (i. e., a statement cannot be inferred to be false based on failures to prove it) and under the absence of the Unique Name Assumption (UNA) (i. e., two different names may refer to the same object). These two features make it difficult to validate data in applications where data is supposed to be complete. Definitions of integrity constraint semantics in OWL using the Closed-World Assumption [103, 104, 144] overcome these issues.

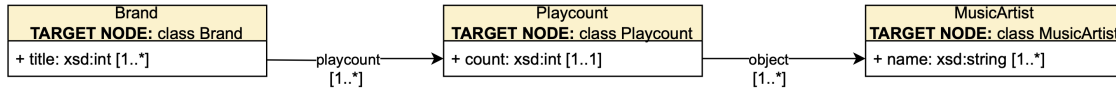
Contrarily, constraints expressed using SPARQL queries or shapes will be evaluated under the Closed-World Assumption (CWA) and following the Unique Name Assumption (UNA). Nevertheless, some constraints may be difficult to express in SPARQL, and the specification process is prone to errors and difficult to maintain.

Data quality conditions and integrity constraints can also be expressed as graphs of shapes or the so-called *shapes schema*. A shape corresponds to a conjunction of constraints that a set of nodes in an RDF graph must satisfy [79]. These constraints can restrict the types of nodes, the cardinality of certain properties, and the expected data types or values for specific properties. A shape can target the instances of a class, the domain or range of a property, or a specific node in the RDF graph. A shape or node in a shapes graph is validated in an RDF graph, if and only if, all the target nodes in the RDF graph satisfy all the constraints in the shape. Figure 6 presents a

⁴⁸<https://github.com/carml/carml>

⁴⁹<https://zenodo.org/record/7689310>

shapes graph of three shapes targeting the classes `Brand`, `Playcount`, and `MusicArtist`. Each of the shapes comprises one constraint. In the shapes `Brand` and `MusicArtist` the properties `title` and `name` can take more than one value, while the shape `Playcount` states that each instance of the class `Playcount` must have exactly one value of the property `count`. Additionally, the instances of the class `Brand` must be related to valid instances of the class `Playcount` which should also be related to valid instances of the class `MusicArtist`.



■ **Figure 6 Shapes for Graph Data.** A shapes graph comprises three shapes interlinked by the properties `playcount` and `objects` between the target classes `Brand`, `Playcount`, and `MusicArtist`.

There are two standards for defining shapes, ShEx (Shape Expressions) [117] and SHACL (Shapes Constraint Language) [95]. Both define shapes over the attributes (i. e., `owl:Datatype-Properties`), and constraints on incoming/outgoing arcs, cardinalities, RDF syntax, and extension mechanism. These inter-class constraints induce a shape network used to validate the integrity and data quality properties of an RDF graph.

SHACL and ShEx, although sharing a common goal, adopt distinct approaches. ShEx seeks to offer a language serving as a grammar or schema for RDF graphs, delineating RDF graph structures for validation. On the other hand, SHACL is positioned as the W3C recommendation for validating RDF graphs against a conjunction of constraints, emphasizing a constraint language for RDF. Despite their analogous roles in specifying shapes and constraints for RDF data, ShEx and SHACL differ in syntax, expressiveness, and community adoption [59].

The evaluation results of a SHACL shape network over an RDF graph are presented in validation reports using a controlled vocabulary. A validation report includes explanations about the violations, the severity of the violation, and a message describing the violation. SHACL is the language selected by the International Data Space (IDS) to express the restrictions that state the integrity over RDF graphs [96]. Besides the integrity validation of an RDF graph, SHACL can be utilized to describe data sources and the certification of a query answer [124], as metadata to enhance the performance of a SPARQL query engine [118], to certify access policies [125], and to provide provenance as a result of the validation of integrity constraints [40].

In the context of a quality assessment pipeline, one crucial step involves validating the shape schema against a graph. It is important to mention that the validation of recursive shape schemas is not explicitly addressed in the SHACL specification [95]. To address this gap, Cormann et al. [36] introduce a semantic framework for validating recursive SHACL. They also demonstrated that validating full SHACL features is an NP-hard problem. Building on these insights, they proposed specific fragments of SHACL that are computationally tractable, along with a fundamental algorithm for validating shape schemas using SPARQL [35]. In a related vein, Andresel et al. [7] propose a stricter semantics for recursive SHACL, drawing inspiration from stable models employed in Answer Set Programming (ASP). This innovative approach enables the representation of SHACL constraints as logic programs and leverages existing ASP solvers for shape schema validation. Importantly, this approach allows for the inclusion of negations in recursive validations. Further, Figuera et al. [51] present Trav-SHACL, an approach that focuses on query optimization techniques aimed at enhancing the incremental behavior and scalability of shape schema validation.

While SHACL has been adopted in a broad range of use cases, given a large graph it remains a challenge how to define shapes efficiently [119]. In many industrial settings with billions of entities and facts [108] creating shapes manually simply is not an option. The current state of the art can automatically extract shapes on WikiData (ca. 2 Billion facts) in less than 1.5 hours while

filtering shapes based on the well-established notions of support and confidence to avoid reporting thousands of shapes that are so rare or apply to such a small subset of the data that they become meaningless [120]. Still, more work is needed to increase scalability further and also to help users make good use of the mined shapes [121] and, e. g., interactively use them to correct and improve the quality of their graphs.

6 Reasoning over and Linking of Graph Data

Section 3 introduced several formal languages for knowledge representation on the Semantic Web. RDF allows the description of simple facts (statements with subject, predicate, and object, so-called RDF triples), e. g., “Anni-Frid Lyngstad” “is a member of” “ABBA”. RDFS allows the definition of types of entities (classes), relationships between classes, and a subclass and superclass hierarchy between types (analogously for relations). OWL is even more expressive than RDF and RDFS. For example, OWL allows the definition of disjoint classes or the description of classes in terms of intersection, union, and complement of other classes.

Below, we first introduce the reasoning over RDFS and OWL at the example of our BBC scenario from Section 2. Subsequently, we discuss works on linking data objects and concepts.

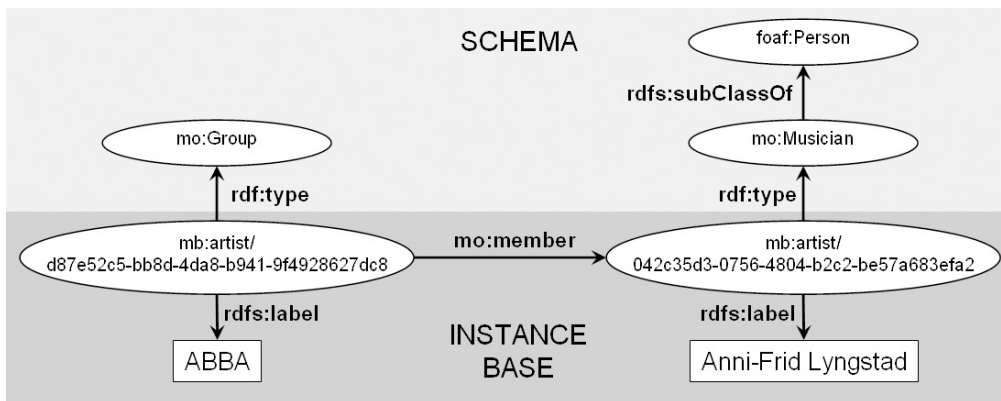
6.1 Reasoning over Graph Data

Based on formal languages representing graph data and their semantics, further (implicit) facts can be derived from the knowledge base by deductive inference. In the following, we exemplify the derivation of implicit facts from a set of explicitly given facts using the RDFS construct `rdfs:subClassOf` and the OWL construct `owl:sameAs`. The property `rdfs:subClassOf` describes hierarchical relationships between classes and with `owl:sameAs` two resources can be defined as identical.

As a first example, we consider the class `foaf:Person`, which is defined in the FOAF ontology, and the classes `mo:Musician` and `mo:Group`, which are defined in the music ontology. In the music ontology, there is an additional axiom that defines `mo:Musician` as a subclass of `foaf:Person` using `rdfs:subClassOf`. Given this axiom, it can be deduced by deductive inference that instances of `mo:Musician` are also instances of `foaf:Person`. Now if there is such a hierarchy of classes and in addition a statement that Anni-Frid Lyngstad is of type `mo:Musician`, then it can be inferred by inference that Anni-Frid Lyngstad is also of type `foaf:Person`. This means that all queries asking for entities of type `foaf:Person` will also include Anni-Frid Lyngstad in the query result, even if that entity is not explicitly defined as an instance of `foaf:Person`. Figure 7 represents these facts and the corresponding class hierarchy in RDFS as a directed graph.

In the second example, the OWL construct `owl:sameAs` is used to define two resources as identical, for example `http://www.bbc.co.uk/music/artists/d87e52c5-bb8d-4da8-b941-9f4928627dc8#artist` and `http://dbpedia.org/resource/ABBA`. Identical here means that these two URIs represent the same real-world object. By inference, information about ABBA from different sources can now be linked. Since ontologies are created independently on the web, and URIs are subject to local naming conventions, a real-world object may be represented by multiple URIs (in different ontologies).

OWL offers a variety of other constructs for the description of classes, relationships, and concrete facts. For example, OWL allows the declaration of transitive relations and inverse relations. For example, the relation “is-member” is inverse to “has-member”. OWL reasoning allows, among other things, consistency checking of an ontology or checking the satisfiability of classes [80]. A class is satisfiable if there can be instances of that class.



■ **Figure 7** Visualization of RDF sample data about ABBA and Anni-Frid Lyngstad to illustrate inference in RDFS.

For a detailed discussion about OWL reasoning, we refer to the literature such as [80, 13]. Different reasoners for OWL have seen widespread adoption in the community such as the well-known Pellet⁵⁰ and Hermit [62]. Finally, a combination of description logic and rules is also possible. For example, Motik et al. [105] presented a combination of description logic and rules that allows tractable inference on OWL ontologies.

6.2 Linking of Objects and Concepts

In the Semantic Web, it cannot be assumed that two URIs refer to two different real-world objects (cf. unique name assumption in Section 5.2). A URI by itself, or in itself, has no identity [70]. Rather, the identity or interpretation of a URI is revealed by the context in which it is used on the Semantic Web. Determining whether or not two URIs refer to the same entity is not a simple task and has been studied extensively in data mining and language understanding in the past. For example, to identify whether or not the author names of research papers refer to the same person, it is often not sufficient to resolve the name, venue, title, and co-authors [90]. The process of determining the identity of a resource is often referred to as entity resolution [90], coreference resolution [156], object identification [123], and normalization [156, 157]. Correctly determining the identity of entities on the Web is important as more and more records appear on the Web and this presents a significant hurdle for very large Semantic Web applications [61].

To address this, a number of services exist that can recognize entities and determine their identity: Thomson Reuters offers OpenCalais⁵¹, a service that can link natural language text to other resources using entity recognition. Another commercial tool that allows for extracting knowledge graphs from text is provided by DiffBot.⁵² Recently, the language model ChatGPT has been compared to the specialized entity and relation extraction tool REBEL [27] for the task of creating knowledge graphs from sustainability-related text [145]. The experiments suggest that large language models improve the accuracy of creating knowledge graphs [145]. The sameAs⁵³ service aims to detect duplicate resources on the Semantic Web using the OWL relationship `owl:sameAs`. This can be used to resolve coreferences between different datasets. For example,

⁵⁰ <https://github.com/stardog-union/pellet>

⁵¹ <https://www.refinitiv.com/en/products/intelligent-tagging-text-analytics>

⁵² <https://www.diffbot.com/>

⁵³ <http://sameas.org/>

for the query with the URI <http://dbpedia.org/resource/ABBA>, a list of over 100 URIs is returned that also references the music group ABBA. One of them is BBC with the resource <http://www.bbc.co.uk/music/artists/d87e52c5-bb8d-4da8-b941-9f4928627dc8#artist>.

Furthermore, the problem of schema matching [157] is very related to the problem of entity resolution, co-reference resolution, and normalization. The goal of schema matching is to address the question of how to integrate data [157], which is non-trivial even for small schemas. In the Semantic Web, schema matching means the matching of different ontologies, respectively the concepts defined in these ontologies. Various (semi-)automatic or machine learning techniques for matching ontologies have been developed in the past [49, 46, 22]. Core ontologies as illustrated in Figure 4.3 represent generic modeling frameworks for integration and alignment with other ontologies. In addition, core ontologies can also integrate Linked Open Data, which typically contains no or very little schema information. The YAGO ontology [140] was generated from the fusion of Wikipedia and Wordnet using rule-based and heuristic methods. A manual assessment showed an accuracy of 95%.

Manual matching of different data sources is also pursued in the Linked Open Data project of the German National Library⁵⁴. For example, the database containing the authors of all documents published in Germany was manually linked with DBpedia and other data sources. A particular challenge was to identify the authors, as described above. For example, former German Chancellor Helmut Kohl has a namesake whose work should not be linked to the chancellor's DBpedia entry. Relationships between keywords used to describe publications are asserted using the SKOS (Simple Knowledge Organization System) vocabulary.⁵⁵ For example, keywords are related to each other using the relation `skos:related`. Hyponyms and hypernyms are expressed by the relations `skos:narrower` and `skos:broader`. Finally, the Ontology Alignment Evaluation Initiative⁵⁶ should be mentioned, which aims to achieve an established consensus for evaluating ontology matching methods.

7 Querying of Linked Data

Queries over Linked Data can be processed using link traversal [74], i. e., the query processor would use one of those IRIs given directly in the query as starting point and query the respective source for more triples involving the IRI. By iteratively doing this for more IRIs and with respect to the graph pattern defined in the query, a local set of triples is collected over which the given query can be evaluated.

More conveniently, queries over RDF and Linked Data can be formulated in SPARQL⁵⁷, if a corresponding endpoint to the graph data is made available. Whereas such queries can target graphs that are stored in a single graph store, Linked Data often requires formulating and executing queries across multiple graphs that are stored at distributed data sources.

Below, we first introduce the basic query processing of SPARQL queries along with our running example. This is followed by discussing RDFS/OWL entailment regimes and querying. Finally, we present approaches for distributed querying over multiple SPARQL endpoints.

⁵⁴ <http://www.d-nb.de/>

⁵⁵ <https://www.w3.org/TR/2009/REC-skos-reference-20090818/>

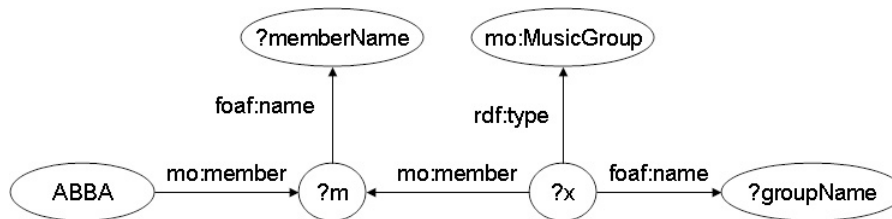
⁵⁶ <http://oaei.ontologymatching.org/>

⁵⁷ <http://www.w3.org/TR/sparql11-query/>

7.1 Basic Query Processing

In principle, a SPARQL query is evaluated by comparing the graph pattern defined in the query to the RDF graph and reporting all matches as results. The set of results can be restricted by additional criteria, such as filters, i. e., conditions on variables and triple patterns that additionally need to be fulfilled.

As an example, let us consider the query illustrated in Figures 8 and 9 that we want to execute over our example MusicBrainz graph from Section 2. We are now interested in the musicians of ABBA who are also members of other bands. If we follow the Linked Data principles and evaluate the query using link traversal [74], this would mean first querying for triples including the IRI that represents ABBA, then navigating to the individual band members, and then following the links to all of the members' bands and query more relevant triples.



■ **Figure 8** Graphical representation of a query for music groups (represented by the variable *?groupName*), whose members are also members of ABBA. The variable *?m* refers to the members of ABBA. The vertex labeled “ABBA” represents the URI for ABBA. The prefix *mo* refers to the music ontology, *foaf* to the FOAF ontology, and *rdf* to the vocabulary of the RDF specification.

```

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX mo: <http://purl.org/ontology/mo/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX bbc: <http://www.bbc.co.uk/music/>
SELECT ?memberName ?groupName
WHERE {
  bbc:artists/d87e52c5-bb8d-4da8-b941-9f4928627dc8#artist mo:member ?m .
  ?x mo:member ?m .
  ?x rdf:type mo:MusicGroup .
  ?m foaf:name ?memberName .
  ?x foaf:name ?groupName }
FILTER (?groupName <> "ABBA")
  
```

■ **Figure 9** SPARQL query for music groups whose members are also members of ABBA. In the first triple pattern of the WHERE part, the URI of ABBA is the subject.

Similar to relational database systems, there exist several dedicated graph stores (aka triple stores) that are optimized for RDF graphs and evaluating SPARQL queries. Some of the most popular triple stores are RDF4J [26], Jena [159], Virtuoso⁵⁸, and GraphDB⁵⁹. They are building upon concepts and techniques known from relational database systems [83, 106] and expand them with graph-specific optimizations [153, 138, 68, 99, 50].

⁵⁸ <http://virtuoso.openlinksw.com>

⁵⁹ <https://graphdb.ontotext.com/>

7.2 Entailment Regimes and Query Processing

In addition to explicitly querying existing facts, SPARQL provides inferencing support through so-called *entailment regimes*. They correspond to logical consequences describing the relationship between the statements that are true when one statement logically follows from one or more statements. Entailment regimes specify an entailment relation between well-formed RDF graphs, assuming that a graph G entails another graph E (denoted $G \models E$) if there is a logical consequence from G to E . A regime extends the query of explicitly existing facts with facts that can be inferred using RDFS and OWL constructs (cf. Section 6), such as the extension of facts about subclasses using `rdfs:subClassOf`.

Depending on the feature set of the respective SPARQL triple store, different (or even no) entailment regimes are supported. They differ in terms of their power in the supported inference capabilities over RDFS/OWL classes and relationships. SPARQL query engines such as GraphDB adopt a materialized approach, wherein they compute the closure of the input RDF graph G over a set of relevant entailment rules R . Conversely, approaches grounded in query rewriting expand the SPARQL query itself rather than altering the RDF graph. Sub-queries are aligned with the entailment rules through backward chaining, and when the consequent of an entailment rule is matched, the antecedent of the rule is added to the query in the form of disjunctions.

Although both approaches yield equivalent answers for a given SPARQL query, their performance can diverge significantly. Materialized RDF query processing may outperform on-the-fly execution of the rewritten query, but it may consume more memory [63]. Nevertheless, various optimization techniques have been proposed to mitigate the overhead caused by the on-the-fly evaluation of entailment regimes [146]. These optimizations are required particularly in the presence of the `owl:sameAs`. This predicate corresponds to logical equivalence and involves the application of the Leibniz Inference Rule [66] to deduce all the equivalent triples entailed by equivalent resources based on `owl:sameAs` relation. This process may lead to many intermediate results, impacting the query engine's performance. Xiao et al. [161] propose query rewriting techniques to efficiently evaluate SPARQL queries with `owl:sameAs` employing equivalent SQL queries.

7.3 Federated Query Processing

Federations provide another perspective on querying linked data over multiple sources. A *federation of knowledge graphs* shares common entities while potentially providing different perspectives on those entities. Each knowledge graph within the federation operates autonomously and can be accessed through various Web interfaces, such as SPARQL endpoints or Linked Data Fragments (LDFs) [151]. SPARQL endpoints offer users the ability to execute any SPARQL query against multiple SPARQL endpoints. In contrast, LDFs enable access to specific graph patterns, such as triple patterns [150] or star-shaped graph patterns [5], allowing retrieval of fragments from an RDF knowledge graph. A *star-shaped subquery* is a conjunction of triple patterns in a SPARQL query that share the same subject variable [153]. An LDF client can submit requests to a server, which then delivers results based on a data shipping policy and partitions results into batches of specified page sizes. Query processing in a federation of graphs differs from querying a single source because it enables real-time data integration of graphs from multiple sources. For example, Figure 10 depicts a SPARQL query whose execution requires the evaluation of subqueries over three knowledge graphs: a Cancer Knowledge Graph (CKG) [6], DBpedia, and Wikidata. This query could not be executed over a single data source unless the three knowledge graphs were physically materialized into one. Subqueries with a specific shape (e.g., star-shaped subqueries) need to be identified and posed against the knowledge graph(s) that is able to answer a particular

part of the query. The federated query engine has to decompose input queries into these subqueries, find a plan to execute them and collect and merge the answers from the subqueries to produce a federated answer.

A federated SPARQL query engine typically follows a mediator and wrapper architecture, which has been established in previous research [158, 162]. Wrappers play a crucial role in translating SPARQL subqueries into requests sent to SPARQL endpoints, while also converting the endpoint responses into internal structures that the query engine can process. The mediator, on the other hand, is responsible for rewriting the original queries into subqueries that can be executed by the data sources within the federation. Additionally, the mediator collects and merges the results obtained from evaluating the subqueries to produce the final answer to the federated query. Essentially, the mediator consists of three main components:

- **Source selection and query decomposition.** This component decomposes queries into subqueries and selects the appropriate graphs (sources) capable of executing each subquery. Simple subqueries typically consist of a list of triple patterns that can be evaluated against at least one graph. Formally, source selection corresponds to the problem of finding the minimal number of knowledge graphs from the federation that can produce a complete answer to the input query. On the other hand, query decomposition requires partitioning the triple patterns of a query into a minimal number of subqueries, such that each subquery can be executed over at least one of the selected knowledge graphs. Commonly, federated query engines follow heuristic-based methods to solve these two problems. For example, for query decomposition, heuristics based on exclusive groups [134] or star-shaped subqueries [153, 152, 102] enable to efficiently solve source selection and query decomposition in queries free of general predicates (e. g., `owl:sameAs` or `rdf:type`).
- **Query optimizer.** This component identifies execution plans by combining star-shaped subqueries (SSQs) and utilizing physical operators implemented by the query engine. Formally, optimizing a query corresponds to the problem of finding a physical plan for the query that minimizes the values of a utility function (e. g., execution time or memory consumption). To maximize the utility function, query optimizers consider plans with different orders of executing operators, alternative implementations of operators, such as joins, as well as particular execution alternatives for certain query types, e. g., queries involving aggregation [86]. In general, finding an optimal solution is computationally intractable [85], while the problems of constructing a *bushy tree plan* [132] and finding an optimal query decomposition over the graphs [152] are NP-Hard. A bushy tree plan is a query execution plan that represents a query as a tree structure with multiple branches or subqueries, which can also be bushy-tree plans. Query plans can be generated following the traditional optimize-then-execute paradigm or re-optimize and adapt a plan on the fly according to the conditions and availability of selected graphs [47]. Alternatively, the query optimizer may resort to a cost model to guide the search on the space of query plans and identify the one that minimizes the values of the utility function [102].
- **Query engine.** This component of a federated query engine implements the physical operators necessary to combine tuples obtained from the graphs. These physical operators are designed to support logical SPARQL operations such as JOIN, UNION, or OPTIONAL [115]. Physical operators can be empowered to adapt execution schedulers to the current conditions of a group of selected graphs. Thus, adaptivity can be achieved at the intra-operator level, where the operators can detect when graphs become blocked or data traffic bursts. Additionally, intra-operator opportunistically produce results as quickly as data arrives from the graphs, and can produce results incrementally. Some opportunistic approaches [4, 82, 56, 3] combine producing results quickly in an incremental fashion with greedy source selection so that the system stops querying additional graphs once the user's wishes, e. g., in terms of the minimum number of obtained results, are fulfilled.

3:24 Semantic Web: Past, Present, and Future

During the query optimization process, a plan is generated as a bushy tree that comprises four join operators. This is shown in Figure 10.

```

PREFIX ex: <http://http://example.com/vocab/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX wdt: <http://www.wikidata.org/prop/direct/>
PREFIX dbp: <http://dbpedia.org/property/>
PREFIX owl: <http://www.w3.org/2002/07/owl#>

```

```

SELECT DISTINCT ?drug ?excretion ?metabolism ?routes ?acting ?mass
WHERE {

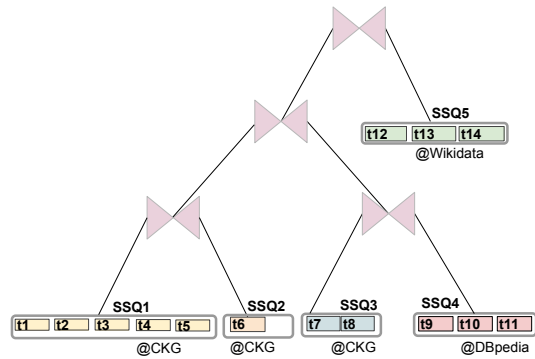
```

t1	?patient rdf:type ex:CPatient .
t2	?patient ex:hasBio ex:EGFR .
t3	?patient ex:hasSmokingHabit ex:NonSmoker .
t4	?patient ex:sex ex:Female .
t5	?patient ex:hasTreatmentEpisode ?treatment .
t6	?treatment ex:hasDrug ?drug .
t7	?drug owl:sameAs ?drug1 .
t8	?drug owl:sameAs ?drug2 .
t9	?drug1 dbp:excretion ?excretion .
t10	?drug1 dbp:metabolism ?metabolism .
t11	?drug1 dbp:routesOfAdministration ?routes .
t12	?drug2 wdt:P592 ?idDrug .
t13	?drug2 wdt:P3780 ?acting .
t14	?drug2 wdt:P2067 ?mass .

```

}
```

(a) Federated Query



(b) Bushy-Tree Plan

Figure 10 Federated query. a) SPARQL query comprising 14 triple patterns to be executed over a federation including Cancer Knowledge Graph (CKG), DBpedia, and Wikidata. b) A query plan composed of five star-shaped subqueries SSQ1, SSQ2, SSQ3, SSQ4, and SSQ5 corresponding to the query decomposition. Each SSQ is executed over the graph that can answer the SSQ. The execution engine follows the query plan; the execution of four joins merges the SSQ answers and produces the federated query answer.

8 Trustworthiness and Provenance of Graph Data

Trustworthiness of web pages and data on the web can be detected by various indicators, e. g., by certificates, by the placement of search engine results, and by links (forward and backward links) to other pages. However, on the Semantic Web, there are few ways for users to assess the trustworthiness of individual data. Rules can be utilized to define policies and business logic over the web of data, and transparently used to infer data that validate or do not validate these policies. The trustworthiness of inferred data can be assessed through its provenance, which encompasses metadata detailing how the data was acquired and verified [94].

The trustworthiness of data on the web can be inferred from the trustworthiness of other users (“Who said that?”), the temporal validity of facts (“When was a fact described?”), or in terms of uncertainty of statements (“To what degree is the statement true?”). Artz and Gil [9] summarize trustworthiness as follows: “Trust is not a new research topic in computer science, spanning areas as diverse as security and access control in computer networks, reliability in distributed systems, game theory and agent systems, and policies for decision-making under uncertainty. The concept of trust in these different communities varies in how it is represented, computed, and used.” Although trustworthiness has long been considered in these areas, the provision and publication of data by many users to multiple sources on the Semantic Web introduces new and unique challenges.

One way of facilitating trust on the Semantic Web is to capture and provide the provenance of data with the PROV ontology (PROV-O)⁶⁰. It captures information about which *Agents* cause data *Entities* to be processed by which *Activities*. Capturing such information requires the use of known tools for modeling metadata for RDF data, e. g., reification, singleton properties, named graphs, or RDF-star⁶¹. While some approaches use these constructs to capture provenance information for each triple individually [54], others exploit the fact that typically multiple triples share the same provenance [72] so that they can be combined into the same named graph encoding the provenance information only once for a set of triples. Delva et al. [40] introduce the notion of shape fragments, which entail the validation of a given shape through the neighborhood of a node, along with the node's provenance and the rationale behind its validation.

Furthermore, trustworthiness also plays a role in inference services on the Semantic Web, as data inference must consider specifications related to trustworthiness and data must be evaluated for trustworthiness. Important aspects for trustworthiness of data include [9]: the origin of the data, trust already gained based on previous interactions, ratings assigned by policies of a system, and access controls and, in some cases, security and importance of information. These aspects are realized in different systems.

In general, data provenance and trustworthiness of data on the Semantic Web have been addressed for RDF data [43, 52] as well as for OWL and rules in [43]. In addition, there are some recent approaches on supporting how-provenance for SPARQL queries [60, 77, 53] with the goal of providing users with explanations on how the answers to their queries were derived from the underlying graphs. Other work deals with access controls over distributed data on the Semantic Web [58]. Furthermore, there are approaches to computing trust values [139] and informativeness of subgraphs [91]. There are also digital signatures for graphs [16]. Analogous to digital signatures for documents, entire graphs or selected vertices and edges of a graph are provided with a digital signature to ensure the authenticity of the data and thus detect unauthorized modifications [92]. In the approach for digital graph signatures developed by Kasten et al., graph data on the Web is supported in RDF format as well as in OWL [93]. The digital graph signature is itself represented as a graph again and can thus be published together with the data on the Web. The link between the signature graph and the signed graph is established by the named graph mechanism [93], although other mechanisms are also possible. Through this mechanism, it is possible to combine and nest signed graphs. It is thus possible to re-sign already signed graphs together with other, new graph data, etc. This makes it possible to build complex chains of trust between publishers of graph data and to be able to prove the origin of data [93, 92].

9 Applications

With the increasing spread and use of semantic and linked data on the Web, the requirements for Semantic Web applications have increased at the same time as their application possibilities. The general requirements for applications based on semantic data on the Web are given by their flexible and diverse representation and descriptions. Applications that use data from relational databases or XML documents can start from a fixed schema. However, this cannot be assumed for data on the Web. Often, neither the data sources nor the type and amount of data in a source are fully known. The dynamics of semantic data on the Web must be taken into account by applications accordingly, both when querying and aggregating data, and when visualizing data. Thus, the real challenge of Semantic Web applications is to guarantee the best possible flexibility of the application to take into account the dynamics of data sources, data, and schemas during input, processing, and output.

⁶⁰<https://www.w3.org/TR/prov-o/>

⁶¹<https://w3c.github.io/rdf-star/>

In the following, selected examples of Semantic Web applications or application areas are presented. They illustrate how flexibility and quality of search, integration, aggregation, and presentation of data from the Web can be implemented. At the same time, they show the potential of Semantic Web applications. First, uniform vocabularies and schemas are presented using the example of *schema.org*. These serve as a basis for semantic search to provide search engines with information about the meaning of web document content. The search and integration of data from different sources is supported by *Sig.ma*, a semantic web browser. Other applications provide semantic search through other representation formalisms, e. g., *Knowledge Graphs*). Subsequently, the *Facebook Graph-API*, an application programming interface (*API*) to the Facebook (Knowledge) Graph, is introduced.

9.1 Vocabularies and Schemas: Schema.org

In HTML documents, the structure and composition of pages can be described with tags, but not the meaning of the information. Vocabulary, schemas, and microdata can be used as mark-up in HTML documents to describe information about page content and its meaning in a way that search engines can process this information.

Schema.org⁶² is a collection of vocabularies and schemas to enrich HTML pages with additional information. The vocabulary of *Schema.org* includes a set of classes and their properties. A universal class “thing” is the most general, which is a kind of umbrella term for all classes. Other common classes are *Organization*, *Person*, *Event*, and *Place*. Properties are used to describe classes in more detail. For example, a person has the properties such as name, address, and date of birth.

In addition to vocabularies, Schema.org also specifies the use of HTML microdata, with the goal of representing data in HTML documents in as unambiguous a form as possible so that search engines can interpret it correctly. An example of this is formats for unique dates and times, which can also describe intervals to indicate the duration of events.

Schema.org is supported by the search engines Bing, Google, and Yandex, among others. There are extensions and libraries for various programming languages, including PHP, JavaScript, Ruby, and Python, to create web pages and web applications using vocabularies and microdata from Schema.org. Likewise, there are mappings from Schema.org vocabularies and microdata to RDFS.

9.2 Semantic Search

A classic web browser enables the display of web pages. A semantic web browser goes one step further by additionally allowing the user to visualize the underlying information of individual pages, for example in the form of RDF metadata. Semantic Web browsers are also referred to as hyperdata browsers because they allow navigation between data while also allowing one to explore the connection to information about that data. Thus, ordinary users can use and exploit Semantic Web data for their information search.

Sig.ma [147] was an application for (browsing) Semantic Web data, which may come from multiple distributed data sources. Sig.ma provided an API for automatically integrating multiple data sources on the Web. The requested data sources describe information in RDF. A search in Sig.ma was initiated by a textual query from the user. Entities such as people, places, or products can be searched for. Results of a query are presented in aggregated form, that is, properties of the searched entity, such as a person, are presented in aggregated form from different data sources.

⁶²<http://schema.org>

For example, in a person search, information such as e-mail address, address, or current employer can be displayed. In addition to the actual information, links to the underlying data sources are also displayed to allow users to navigate to refine their search. Sigma also supported structured queries in which specific characteristics can be requested for an entity, such as contact information for a specific person.

Queries to data sources occur in parallel. The results from each data source in the form of RDF graphs are summarized by using properties of links in RDF data, such as `owl:sameAs`, or inverse-functional predicates. When searching data sources, techniques such as indexes, logical inference, and heuristics are used for data aggregation. OntoBroker⁶³ [39] and OntoEdit [142] are ontology editors with search and inference systems for ontologies. Using OntoBroker, complex queries over distributed Semantic Web resources, e. g., represented in OWL, RDF, RDFS, SPARQL, and also F-Logic) can be efficiently processed.

9.3 Knowledge Graphs and Wikidata

There is an increasing number of knowledge bases and representations of structured data. For example, the secondary database Wikidata⁶⁴ [154]. A secondary database includes, in addition to the (actual) statements, relationships to their sources and other databases (called secondary information). Wikidata is a shared database between Wikipedia and Wikimedia. Wikidata mainly contains a collection of objects, which are represented as triples over the objects' properties and the corresponding values. Semantic MediaWiki⁶⁵ is an extension of MediaWiki. It serves as a flexible knowledge base and knowledge management system. Semantic MediaWiki extends a classic wiki with the ability to enrich content in a machine-readable way using semantic annotations.

Another knowledge base was Freebase⁶⁶, also an open and collaborative platform initiated in 2007 and acquired by Google in 2010. The content from Freebase was taken from various sources, including parts from the MusicBrainz ontology mentioned earlier. The success and widespread use of Wikidata prompted Google to migrate Freebase to Wikidata [143]. This strengthened the goal to develop a comprehensive, collaborative basis of structured data.

Google offers a semantic search function with Google Knowledge Graph^{67,68}. A knowledge graph, like an RDF graphs, is a set of triples representing links between entities. This forms a semantic database. Possible entity types are described on `schema.org`, among others. If a search term occurs in a query, the corresponding entity is searched for in the knowledge graph. Starting from this entity, it is then possible to navigate to further entities by means of the links.

9.4 API-Access to Social Networks

A social network is essentially a graph in which connections are formed from users to other users, e. g., in the form of a friendship relationship or to events and groups. Facebook's Graph API describes a programming interface to the Facebook Graph (called *Open Graph*). Within the graph, people, events, pages, and photos are represented as objects, with each object having a unique identifier. For example, `https://graph.facebook.com/abba` is the identifier of ABBA's Facebook page. There are also unique identifiers for the possible relationship types of an object, which allow navigating from one object to all connected objects with respect to a particular relationship.

⁶³ <https://www.semafora-systems.com/ontobroker-and-ontostudio-x>

⁶⁴ <https://www.wikidata.org/wiki/Wikidata:Introduction/de>

⁶⁵ https://www.semantic-mediawiki.org/wiki/Semantic_MediaWiki

⁶⁶ <https://www.freebase.com>

⁶⁷ <http://www.google.com/insidesearch/features/search/knowledge.html>

⁶⁸ <https://developers.google.com/knowledge-graph/>

The Graph API allows one to navigate the Facebook Graph and read objects, including their properties and relationships to other objects, as well as creating new objects in the Facebook Graph and deploying applications. The API also supports requests for an object's metadata, such as *when* and *by whom* an object was created.

10 Impact for Practitioners

Linking and using graph data on the Web has become a widespread practice. Today, there is a large amount of open data in various formats and domains, such as bibliographic information management, bioinformatics, and e-government. DBpedia is the central hub in this context, around which different datasets and domains are grouped (cf. [21]). This is illustrated, e. g., by the tremendous growth of the Linked Open Data Cloud⁶⁹ since 2007. Two of the latest notable supporters of graph-based data are online auctioneer eBay with their graph database⁷⁰ and the U.S. space agency NASA with the unification of internal distributed case databases as knowledge graphs⁷¹. These and other success stories of the Semantic Web in industries and industry-scale knowledge graphs are described by Noy et al. [107]. Further analyses and surveys arguing about the importance but also challenges of using graph data can be found in the literature like the 2020 survey of Sahu et al. [127] and the 2021 reflection about the future of graphs by Sakr et al. [128]. The usefulness of knowledge graphs and semantic-based data modeling for complex systems is also discussed in the 2024 book by Abonyi et al. [1].

Regarding lightweight open graph data, Schema.org defines schemas for modeling data on web pages to provide information about the underlying data structures and meaning of the data. Search engines can use this additional information to better analyze the content of web pages. As mentioned above, Schema.org is supported by search engines such as Bing, Google, and Yandex. Studies on selected sources have shown that web pages among the top 10 results have up to 15 % higher click-through rate⁷². Other companies like BestBuy.com even report up to 30 % higher click-through rates since adding semantic data to their websites (cf. Section 9) in 2009. BestBuy.com uses the GoodRelations vocabulary⁷³ to describe online offers. Similarly, Google uses semantic data from online commerce portals that use the GoodRelations vocabulary and takes it into account when searching⁷⁴.

Another success is the publication of government data. For example, the U.S. government makes government data publicly available with data.gov⁷⁵, and U.S. Census⁷⁶ publishes statistical data about the United States. In the UK, data.gov.uk⁷⁷ is a key part of a program to increase data transparency in the public sector. The European Commission operates data.europa.eu⁷⁸, a European data portal with metadata about the member states. Among others, it provides a SPARQL endpoint to access the data.

⁶⁹The growth of the Linked Open Data Cloud is documented at: <http://linkeddata.org/>.

⁷⁰<https://github.com/eBay/akutan>

⁷¹<https://blog.nuclino.com/why-nasa-converted-its-lessons-learned-database-into-a-knowledge-graph>

⁷²<http://developer.yahoo.net/blog/archives/2008/07/>

⁷³<http://www.heppnetz.de/projects/goodrelations/>

⁷⁴<http://www.ebusiness-unibw.org/wiki/GoodRelationsInGoogle>

⁷⁵<http://www.data.gov/>

⁷⁶<http://www.rdfabout.com/demo/census/>

⁷⁷<http://data.gov.uk>

⁷⁸<https://data.europa.eu/>

Finally, a strong growth of semantic biomedical data on the Web can be noted. As part of Bio2RDF⁷⁹, many bioinformatics databases have been linked. Transinsight GmbH offers the knowledge-based search engine GoPubMed⁸⁰ to find biomedical research articles. Ontologies are used for searching.

Regarding more heavyweight ontologies in OWL, there has also been movement in recent years. In addition to numerous research-derived inference engines such as Pellet and Hermit mentioned above, inference mechanisms for OWL can now be found in commercial graph databases such as neo4j⁸¹. Furthermore, pattern-based core ontologies can also be found in software development workflows [133]. The development and use of core ontologies is part of a continuous delivery process that is used in practice.

11 Summary and Outlook

The Semantic Web consists of a variety of techniques that have been heavily influenced by long-term artificial intelligence research and its results. The current state is also driven by an industry uptake under the umbrella term of Knowledge Graphs and reflected in various activities as described. In summary, therefore, it can be observed that semantic data on the Web is having a real impact on commercial providers of products and services, as well as on governments and public administrations.

Despite all the research and industrial developments, the full potential of the Semantic Web has not yet been exploited. Some important components of the Semantic Web architecture are still being explored, such as data provenance and trustworthiness. Below, we describe three example directions for future work.

- Neuro-symbolic systems: As mentioned in the introduction, we see as an important direction of future work the combination of symbolic AI and subsymbolic AI. By combining the strength of Large Language Models (LLM), i. e., generative AI, in processing and generating natural language text and accessing structured data and logical reasoning capabilities of the Semantic Web, a next step towards the vision of automated agents that perform complex planning tasks may be reached. An example is performing A* search with an LLM [164]. Specifically, LLMs might comprehensively capture and acquire human knowledge [41], but current LLMs lack responding to simple questions of non-existing facts in their training data [41], may not contain all facts [141], and thus return less accurate answers [84]. To leverage the distinct capabilities of both LLMs and the Semantic Web, the integration of neuro-symbolic systems appears to offer a viable solution [113]. Neuro-symbolic systems could also address the problem that LLMs' output is based on the most probable answer, which sometimes leads to wrong answers – often referred to as “hallucinations” [14, 81, 141].
- Natural interfaces between machine and users: A key to successful applications of the Semantic Web is intuitive user interfaces. Users must be offered applications that are intuitive and easy to use. This includes improving interfaces based on natural language for formulating queries and accessing structured data stored in SPARQL endpoints. Again, the use and deeper integration of LLMs with Knowledge Graphs shows a promising direction.
- Semantic Web components: There are still components of the architecture (see Section 3) where active development and research are conducted. Most notably, there are crypto and trust. Recent new W3C standards such as DID and Verifiable Credentials have been developed. However, one can expect more work and development in this direction.

⁷⁹ <http://bio2rdf.org/>

⁸⁰ <http://www.gopubmed.org/>

⁸¹ <https://neo4j.com/blog/neo4j-rdf-graph-database-reasoning-engine/>

Finally, we like to point to existing literature discussing the future directions of research on the Semantic Web [30] and Knowledge Graphs [44]. Breit et al. [25] conducted a survey on the fusion of Semantic Web and Machine Learning, exploring the opportunities arising from the convergence of these two paradigms.

References

- 1 János Abonyi, László Nagy, and Tamás Ruppert, editors. *Ontology-Based Development of Industry 4.0 and 5.0 Solutions for Smart Manufacturing and Production: Knowledge Graph and Semantic Based Modeling and Optimization of Complex Systems*. Springer, 2024.
- 2 Ghadeer Abuoda, Christian Aebeloe, Daniele Dell’Aglia, Arthur Keen, and Katja Hose. StarBench: Benchmarking RDF-star Triplestores. In *QuWeDa icw. ISWC 2023*, volume 3565 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2023. URL: <https://ceur-ws.org/Vol-3565/QuWeDa2023-paper4.pdf>.
- 3 Maribel Acosta and Maria-Esther Vidal. Networks of linked data eddies: An adaptive web query processing engine for RDF data. In *The Semantic Web - ISWC 2015 - 14th International Semantic Web Conference, Bethlehem, PA, USA, October 11-15, 2015, Proceedings, Part I*, volume 9366 of *Lecture Notes in Computer Science*, pages 111–127. Springer, 2015. doi:10.1007/978-3-319-25007-6_7.
- 4 Maribel Acosta, Maria-Esther Vidal, Tomas Lampo, Julio Castillo, and Edna Ruckhaus. ANAPSID: an adaptive query processing engine for SPARQL endpoints. In Lora Aroyo, Chris Welty, Harith Alani, Jamie Taylor, Abraham Bernstein, Lalana Kagal, Natasha Fridman Noy, and Eva Blomqvist, editors, *The Semantic Web - ISWC 2011 - 10th International Semantic Web Conference, Bonn, Germany, October 23-27, 2011, Proceedings, Part I*, volume 7031 of *Lecture Notes in Computer Science*, pages 18–34. Springer, 2011. doi:10.1007/978-3-642-25073-6_2.
- 5 Christian Aebeloe, Ilkcan Keles, Gabriela Montoya, and Katja Hose. Star pattern fragments: Accessing knowledge graphs through star patterns. *CoRR*, abs/2002.09172, 2020. doi:10.48550/arXiv.2002.09172.
- 6 Fotis Aisopos, Samaneh Jozashoori, Emetis Niazmand, Disha Purohit, Ariam Rivas, Ahmad Sakor, Enrique Iglesias, Dimitrios Vogiatzis, Ernestina Menasalvas, Alejandro Rodríguez González, Guillermo Viguera, Daniel Gómez-Bravo, Maria Torrente, Roberto Hernández López, Mariano Provencio Pulla, Athanasios Dalianis, Anna Triantafyllou, Georgios Paliouras, and Maria-Esther Vidal. Knowledge graphs for enhancing transparency in health data ecosystems. *Semantic Web*, 14(5):943–976, 2023. doi:10.3233/SW-223294.
- 7 Medina Andreeşel, Julien Corman, Magdalena Ortiz, Juan L. Reutter, Ognjen Savković, and Mantas Šimkus. Stable model semantics for recursive SHACL. In *ACM-The Web Conference*, pages 1570–1580, 2020. doi:10.1145/3366423.3380229.
- 8 Julián Arenas-Guerrero, David Chaves-Fraga, Jhon Toledo, María S. Pérez, and Oscar Corcho. Morph-KGC: Scalable knowledge graph materialization with mapping partitions. *Semantic Web*, 15(1), 2022. doi:10.3233/SW-223135.
- 9 Donovan Artz and Yolanda Gil. A Survey of Trust in Computer Science and the Semantic Web. *J. Web Sem.*, 5(2):58–71, 2007. doi:10.1016/j.websem.2007.03.002.
- 10 Dylan Van Assche, Thomas Delva, Gerald Haesendonck, Pieter Heyvaert, Ben De Meester, and Anastasia Dimou. Declarative RDF graph generation from heterogeneous (semi-)structured data: A systematic literature review. *J. Web Semant.*, 75:100753, 2023. doi:10.1016/j.websem.2022.100753.
- 11 S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives. DBpedia: A Nucleus for a Web of Open Data. In *Semantic Web Conference and Asian Semantic Web Conference*, pages 722–735, nov 2008. doi:10.1007/978-3-540-76298-0_52.
- 12 Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.
- 13 Franz Baader and Ulrike Sattler. An overview of tableau algorithms for description logics. *Stud Logica*, 69(1):5–40, 2001. doi:10.1023/A:1013882326814.
- 14 Yejin Bang, Samuel Cahyawijaya, Nayeon Lee, Wenliang Dai, Dan Su, Bryan Wilie, Holy Lovenia, Ziwei Ji, Tiezheng Yu, Willy Chung, Quyet V. Do, Yan Xu, and Pascale Fung. A Multitask, Multilingual, Multimodal Evaluation of ChatGPT on Reasoning, Hallucination, and Interactivity. *CoRR*, abs/2302.04023, 2023. doi:10.48550/arXiv.2302.04023.
- 15 David Beckett, Tim Berners-Lee, Eric Prud’hommeaux, and Gavin Carothers. Terse RDF Triple Language, 2014. <http://www.w3.org/TR/turtle/>.
- 16 Luigi Bellomarini, Markus Nissl, and Emanuel Sallinger. Blockchains as knowledge graphs - blockchains for knowledge graphs (vision paper). In *Proceedings of the International Workshop on Knowledge Representation and Representation Learning co-located with the 24th European Conference on Artificial Intelligence (ECAI 2020), Virtual Event, September, 2020*, volume 3020 of *CEUR Workshop Proceedings*, pages 43–51. CEUR-WS.org, 2020. URL: https://ceur-ws.org/Vol-3020/KR4L_paper_3.pdf.
- 17 T. Berners-Lee. Universal Resource Identifiers in WWW: A Unifying Syntax for the Expression of Names and Addresses of Objects on the Network

- as used in the World-Wide Web. RFC 1630, Internet Engineering Task Force, jun 1994. URL: <http://www.rfc-editor.org/rfc/rfc1630.txt>.
- 18 Tim Berners-Lee, Roy T. Fielding, and Larry M Masinter. Uniform Resource Identifier (URI): Generic Syntax. Technical Report 3986, jan 2005. doi:10.17487/RFC3986.
 - 19 Tim Berners-Lee, James Hendler, and Orla Lassila. The Semantic Web. *Scientific American*, pages 1–4, 2001.
 - 20 Tim Berners-Lee, Larry M Masinter, and Mark P. McCahill. Uniform Resource Locators (URL). Technical Report 1738, dec 1994. doi:10.17487/RFC1738.
 - 21 Christian Bizer. The Emerging Web of Linked Data. *IEEE Intelligent Systems*, 24(5):87–92, 2009. doi:10.1109/MIS.2009.102.
 - 22 Eva Blomqvist. Ontocase-automatic ontology enrichment based on ontology design patterns. In *International Semantic Web Conference*, pages 65–80, 2009. doi:10.1007/978-3-642-04930-9_5.
 - 23 Till Blume. *Semantic structural graph summaries for evolving and distributed graphs*. PhD thesis, University of Ulm, Germany, 2022. URL: <https://nbn-resolving.org/urn:nbn:de:bsz:289-oparu-46050-1>.
 - 24 Stefano Borgo and Claudio Masolo. Foundational choices in DOLCE. In *Handbook on Ontologies*, page 361–381. Springer, 2nd edition, 2009. doi:10.1007/978-3-540-92673-3_16.
 - 25 Anna Breit, Laura Waltersdorfer, Fajar J. Ekaputra, Marta Sabou, Andreas Ekelhart, Andreea Iana, Heiko Paulheim, Jan Portisch, Artem Revenko, Annette Ten Teije, and Frank Van Harmelen. Combining machine learning and semantic web: A systematic mapping study. *ACM Comput. Surv.*, 55(14s), jul 2023. doi:10.1145/3586163.
 - 26 Jeen Broekstra, Arjoun Kampman, and Frank Van Harmelen. Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema. In *International Semantic Web Conference*, pages 54–68. Springer, 2002. doi:10.1007/3-540-48005-6_7.
 - 27 Pere-Lluís Huguet Cabot and Roberto Navigli. REBEL: relation extraction by end-to-end language generation. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih, editors, *Findings of the Association for Computational Linguistics: EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 16-20 November, 2021*, pages 2370–2381. Association for Computational Linguistics, 2021. doi:10.18653/v1/2021.FINDINGS-EMNLP.204.
 - 28 Diego Calvanese, Benjamin Cogrel, Sarah Komla-Ebri, Roman Kontchakov, Davide Lanti, Martin Rezk, Mariano Rodriguez-Muro, and Guohui Xiao. Ontop: Answering SPARQL queries over relational databases. *Semantic Web*, 8(3):471–487, 2017. doi:10.3233/SW-160217.
 - 29 Jeremy J. Carroll, Christian Bizer, Patrick J. Hayes, and Patrick Stickler. Semantic web publishing using named graphs. In Jennifer Golbeck, Piero A. Bonatti, Wolfgang Nejdl, Daniel Olmedilla, and Marianne Winslett, editors, *Proceedings of the ISWC*04 Workshop on Trust, Security, and Reputation on the Semantic Web, Hiroshima, Japan, November 7, 2004*, volume 127 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2004. URL: <https://ceur-ws.org/Vol-127/paper2.pdf>.
 - 30 Irene Celino and Heiko Paulheim. The time traveler’s guide to semantic web research: Analyzing fictitious research themes in the ESWC "next 20 years" track. *CoRR*, abs/2309.13939, 2023. doi:10.48550/arXiv.2309.13939.
 - 31 David Chaves-Fraga, Kemele M. Endris, Enrique Iglesias, Óscar Corcho, and Maria-Esther Vidal. What are the parameters that affect the construction of a knowledge graph? In *On the Move to Meaningful Internet Systems: OTM 2019 Conferences - Confederated International Conferences: CoopIS, ODBASE, C&TC 2019, Rhodes, Greece, October 21-25, 2019, Proceedings*, volume 11877 of *Lecture Notes in Computer Science*, pages 695–713. Springer, 2019. doi:10.1007/978-3-030-33246-4_43.
 - 32 David Chaves-Fraga, Edna Ruckhaus, Freddy Priyatna, Maria-Esther Vidal, and Óscar Corcho. Enhancing virtual ontology based access over tabular data with Morph-CSV. *Semantic Web*, 12(6):869–902, 2021. doi:10.3233/SW-210432.
 - 33 Michelle Cheatham and Catia Pesquita. *Semantic Data Integration*, pages 263–305. Springer International Publishing, Cham, 2017. doi:10.1007/978-3-319-49340-4_8.
 - 34 Artem Chebotko, Shiyong Lu, and Farshad Fotouhi. Semantics preserving SPARQL-to-SQL translation. *Data Knowl. Eng.*, 68(10):973–1000, 2009. doi:10.1016/j.datak.2009.04.001.
 - 35 Julien Corman, Fernando Florenzano, Juan L. Reutter, and Ognjen Savković. Validating SHACL Constraints over a SPARQL Endpoint. In *International Semantic Web Conference ISWC*, pages 145–163. Springer, 2019. doi:10.1007/978-3-030-30793-6_9.
 - 36 Julien Corman, Juan L. Reutter, and Ognjen Savković. Semantics and Validation of recursive SHACL. In *International Semantic Web Conference ISWC*, pages 318–336. Springer, 2018. doi:10.1007/978-3-030-00671-6_19.
 - 37 Souripriya Das, Seema Sundara, and Richard Cyganiak. R2RML: RDB to RDF Mapping Language, 2012. <https://www.w3.org/TR/r2rml/>.
 - 38 Jeremy Debattista, Christoph Lange, Sören Auer, and Dominic Cortis. Evaluating the quality of the LOD cloud: An empirical investigation. *Semantic Web*, 9(6):859–901, 2018. doi:10.3233/SW-180306.
 - 39 Stefan Decker, Michael Erdmann, Dieter Fensel, and Rudi Studer. Ontobroker: Ontology based access to distributed and semi-structured information. In *Database Semantics - Semantic Issues in Multimedia Systems, IFIP TC2/WG2.6 Eighth Working Conference on Database Semantics (DS-8), Rotorua, New Zealand, January 4-8, 1999*, pages 351–369. Kluwer, 1999.
 - 40 Thomas Delva, Anastasia Dimou, Maxime Jakubowski, and Jan Van den Bussche. Data provenance for SHACL. In *Proceedings 26th International Conference on Extending Database Technology, EDBT 2023, Ioannina, Greece, March 28-31, 2023*, pages 285–297. OpenProceedings.org, 2023. doi:10.48786/edbt.2023.23.

- 41 Peter J. Denning. The smallness of large language models. *Commun. ACM*, 66(9):24–27, aug 2023. doi:10.1145/3608966.
- 42 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. In Christian Bizer, Tom Heath, Sören Auer, and Tim Berners-Lee, editors, *Proceedings of the Workshop on Linked Data on the Web co-located with the 23rd International World Wide Web Conference (WWW 2014), Seoul, Korea, April 8, 2014*, volume 1184 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2014. URL: https://ceur-ws.org/Vol-1184/ldow2014_paper_01.pdf.
- 43 Renata Queiroz Dividino, Simon Schenk, Sergej Sizov, and Steffen Staab. Provenance, Trust, Explanations - and all that other Meta Knowledge. *KI*, 23(2):24–30, 2009. URL: http://www.kuenstliche-intelligenz.de/fileadmin/template/main/archiv/pdf/ki2009-02_page24_web_teaser.pdf.
- 44 Xin Luna Dong. Generations of knowledge graphs: The crazy ideas and the business impact. *CoRR*, abs/2308.14217, 2023. doi:10.48550/arXiv.2308.14217.
- 45 M. Duerst and M. Suignard. Internationalized resource identifiers (IRIs). RFC 3987, Internet Engineering Task Force, jan 2005. URL: <http://www.rfc-editor.org/rfc/rfc3987.txt>.
- 46 Marc Ehrig. *Ontology Alignment: Bridging the Semantic Gap*, volume 4 of *Semantic Web and Beyond*. Springer, 2007. doi:10.1007/978-0-387-36501-5.
- 47 Kemele M. Endris, Maria-Esther Vidal, and Damien Graux. Federated query processing. In Valentina Janev, Damien Graux, Hajira Jabeen, and Emanuel Sallinger, editors, *Knowledge Graphs and Big Data Processing*, volume 12072 of *Lecture Notes in Computer Science*, pages 73–86. Springer, 2020. doi:10.1007/978-3-030-53199-7_5.
- 48 Jérôme Euzenat and Pavel Shvaiko. *Ontology matching*, chapter Classifications of ontology matching techniques, pages 61–72. Springer, 2007. doi:10.1007/978-3-540-49612-0.
- 49 Jérôme Euzenat and Pavel Shvaiko. *Ontology matching*. Springer, 2007. doi:10.1007/978-3-540-49612-0.
- 50 Sebastián Ferrada, Benjamin Bustos, and Aidan Hogan. Extending SPARQL with similarity joins. In *ISWC (1)*, volume 12506 of *Lecture Notes in Computer Science*, pages 201–217. Springer, 2020. doi:10.1007/978-3-030-62419-4_12.
- 51 Mónica Figuera, Philipp D. Rohde, and Maria-Esther Vidal. Trav-shacl: Efficiently validating networks of SHACL constraints. In Jure Leskovec, Marko Grobelnik, Marc Najork, Jie Tang, and Leila Zia, editors, *WWW '21: The Web Conference 2021, Virtual Event / Ljubljana, Slovenia, April 19–23, 2021*, pages 3337–3348. ACM / IW3C2, 2021. doi:10.1145/3442381.3449877.
- 52 Giorgos Flouris, Irini Fundulaki, Panagiotis Padiaditis, Yannis Theoharis, and Vassilis Christophides. Coloring RDF Triples to Capture Provenance. In *International Semantic Web Conference*, volume 5823 of *LNCS*, pages 196–212. Springer, 2009. doi:10.1007/978-3-642-04930-9_13.
- 53 Luis Galárraga, Daniel Hernández, Anas Katim, and Katja Hose. Visualizing how-provenance explanations for SPARQL queries. In *WWW (Companion Volume)*, pages 212–216. ACM, 2023. doi:10.1145/3543873.3587350.
- 54 Luis Galárraga, Kim Ahlström Meyn Mathiasen, and Katja Hose. QBOAirbase: The European Air Quality Database as an RDF Cube. In *ISWC (Posters, Demos & Industry Tracks)*, volume 1963 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2017. URL: <https://ceur-ws.org/Vol-1963/paper507.pdf>.
- 55 Mikhail Galkin, Sören Auer, Maria-Esther Vidal, and Simon Scerri. Enterprise knowledge graphs: A semantic approach for knowledge management in the next generation of enterprise information systems. In Slimane Hammoudi, Michal Smialek, Olivier Camp, and Joaquim Filipe, editors, *ICEIS 2017 - Proceedings of the 19th International Conference on Enterprise Information Systems, Volume 2, Porto, Portugal, April 26–29, 2017*, pages 88–98. SciTePress, 2017. doi:10.5220/0006325200880098.
- 56 Mikhail Galkin, Kemele M. Endris, Maribel Acosta, Diego Collarana, Maria-Esther Vidal, and Sören Auer. Smjoin: A multi-way join operator for SPARQL queries. In Rinke Hoekstra, Catherine Faron-Zucker, Tassilo Pellegrini, and Victor de Boer, editors, *Proceedings of the 13th International Conference on Semantic Systems, SEMANTiCS 2017, Amsterdam, The Netherlands, September 11–14, 2017*, pages 104–111. ACM, 2017. doi:10.1145/3132218.3132220.
- 57 Aldo Gangemi and Valentina Presutti. Ontology design patterns. In *Handbook on Ontologies*, page 221–243. Springer, 2009. doi:10.1007/978-3-540-92673-3_10.
- 58 Rita Gavriiloae, Wolfgang Nejdl, Daniel Olmedilla, Kent E. Seamons, and Marianne Winslett. No Registration Needed: How to Use Declarative Policies and Negotiation to Access Sensitive Resources on the Semantic Web. In *European Semantic Web Symposium*, volume 3053 of *LNCS*, pages 342–356. Springer, 2004. doi:10.1007/978-3-540-25956-5_24.
- 59 José Emilio Labra Gayo, Eric Prud'hommeaux, Iovka Boneva, and Dimitris Kontokostas. *Validating RDF Data*. Synthesis Lectures on the Semantic Web: Theory and Technology. Morgan & Claypool Publishers, 2017. doi:10.2200/S00786ED1V01Y201707WBE016.
- 60 Floris Geerts, Thomas Unger, Grigoris Karvounarakis, Irini Fundulaki, and Vassilis Christophides. Algebraic structures for capturing the provenance of SPARQL queries. *J. ACM*, 63(1):7:1–7:63, 2016. doi:10.1145/2810037.
- 61 H. Glaser, A. Jaffri, and I. Millard. Managing co-reference on the semantic web. In *WWW2009 Workshop: Linked Data on the Web*, 2009. URL: https://ceur-ws.org/Vol-538/ldow2009_paper11.pdf.
- 62 Birte Glimm, Ian Horrocks, Boris Motik, Giorgos Stoilos, and Zhe Wang. Hermit: An OWL 2 rea-

- soner. *J. Autom. Reasoning*, 53(3):245–269, 2014. doi:10.1007/s10817-014-9305-1.
- 63 Birte Glimm, Yevgeny Kazakov, Ilianna Kollia, and Giorgos B. Stamou. Lower and upper bounds for SPARQL queries over OWL ontologies. In Blai Bonet and Sven Koenig, editors, *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA*, pages 109–115. AAAI Press, 2015. doi:10.1609/aaai.v29i1.9192.
- 64 Birte Glimm and Heiner Stuckenschmidt. 15 years of semantic web: An incomplete survey. *KI*, 30(2):117–130, 2016. doi:10.1007/s13218-016-0424-1.
- 65 Asunción Gómez-Pérez, Mariano Fernández López, and Oscar Corcho. *Ontological engineering*. Springer, 2004. doi:10.1007/b97353.
- 66 David Gries and Fred B. Schneider. *A Logical Approach to Discrete Math*. Texts and Monographs in Computer Science. Springer, 1993. doi:10.1007/978-1-4757-3837-7.
- 67 Gerd Gröner, Ansgar Scherp, and Steffen Staab. Semantic web. In Günther Görz, Josef Schneeberger, and Ute Schmid, editors, *Handbuch der Künstlichen Intelligenz, 5. Auflage*, pages 585–612. Oldenbourg Wissenschaftsverlag, 2013. doi:10.1524/9783486719796.585.
- 68 Andrey Gubichev and Thomas Neumann. Exploiting the query structure for efficient join ordering in SPARQL queries. In *EDBT*, pages 439–450. OpenProceedings.org, 2014. doi:10.5441/002/edbt.2014.40.
- 69 Ivan Habernal and Miloslav Konopík. SWSNL: semantic web search using natural language. *Expert Syst. Appl.*, 40(9):3649–3664, 2013. doi:10.1016/j.eswa.2012.12.070.
- 70 Harry Halpin and Valentina Presutti. An ontology of resources: Solving the identity crisis. In *European Semantic Web Conference*, pages 521–534, 2009. doi:10.1007/978-3-642-02121-3_39.
- 71 William L. Hamilton. *Graph Representation Learning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2020. doi:10.2200/S01045ED1V01Y202009AIM046.
- 72 Emil Riis Hansen, Matteo Lissandrini, Agneta Ghose, Søren Løkke, Christian Thomsen, and Katja Hose. Transparent Integration and Sharing of Life Cycle Sustainability Data with Provenance. In *ISWC*, volume 12507 of *Lecture Notes in Computer Science*, pages 378–394. Springer, 2020. doi:10.1007/978-3-030-62466-8_24.
- 73 Andreas Harth, Katja Hose, and Ralf Schenkel, editors. *Linked Data Management*. Chapman and Hall/CRC, 2014. URL: <http://www.crcnetbase.com/isbn/9781466582415>.
- 74 Olaf Hartig. Zero-Knowledge Query Planning for an Iterator Implementation of Link Traversal Based Query Execution. In *ESWC (1)*, volume 6643 of *Lecture Notes in Computer Science*, pages 154–169. Springer, 2011. doi:10.1007/978-3-642-21034-1_11.
- 75 Olaf Hartig, Katja Hose, and Juan F. Sequeda. Linked Data Management. In *Encyclopedia of Big Data Technologies*. Springer, 2019. doi:10.1007/978-3-319-63962-8_76-1.
- 76 Jochen Heinsohn, Daniel Kudenko, Bernhard Nebel, and Hans-Jürgen Profitlich. An Empirical Analysis of Terminological Representation Systems. *Artif. Intell.*, 68(2):367–397, 1994. doi:10.1016/0004-3702(94)90071-x.
- 77 Daniel Hernández, Luis Galárraga, and Katja Hose. Computing How-Provenance for SPARQL Queries via Query Rewriting. *Proc. VLDB Endow.*, 14(13):3389–3401, 2021. doi:10.14778/3484224.3484235.
- 78 Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia d’Amato, Gerard de Melo, Claudio Gutierrez, José Emilio Labra Gayo, Sabrina Kirrane, Sebastian Neumaier, Axel Polleres, Roberto Navigli, Axel-Cyrille Ngonga Ngomo, Sabbir M. Rashid, Anisa Rula, Lukas Schmelzeisen, Juan F. Sequeda, Steffen Staab, and Antoine Zimmermann. Knowledge graphs. *CoRR*, abs/2003.02320, 2020. doi:10.48550/arXiv.2003.02320.
- 79 Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia d’Amato, Gerard de Melo, Claudio Gutierrez, Sabrina Kirrane, José Emilio Labra Gayo, Roberto Navigli, Sebastian Neumaier, Axel-Cyrille Ngonga Ngomo, Axel Polleres, Sabbir M. Rashid, Anisa Rula, Lukas Schmelzeisen, Juan Sequeda, Steffen Staab, and Antoine Zimmermann. *Knowledge Graphs*. Synthesis Lectures on Data, Semantics, and Knowledge. Morgan & Claypool Publishers, 2021. doi:10.2200/S01125ED1V01Y202109DSK022.
- 80 Ian Horrocks and Peter F. Patel-Schneider. KR and reasoning on the semantic web: OWL. In John Domingue, Dieter Fensel, and James A. Hendler, editors, *Handbook of Semantic Web Technologies*, pages 365–398. Springer, 2011. doi:10.1007/978-3-540-92913-0_9.
- 81 Katja Hose. Knowledge engineering in the era of artificial intelligence. In *ADBIS*, volume 13985 of *Lecture Notes in Computer Science*, pages 3–15. Springer, 2023. doi:10.1007/978-3-031-42914-9_1.
- 82 Katja Hose and Ralf Schenkel. Towards benefit-based RDF source selection for SPARQL queries. In *SWIM*, page 2. ACM, 2012. doi:10.1145/2237867.2237869.
- 83 Katja Hose, Ralf Schenkel, Martin Theobald, and Gerhard Weikum. Database Foundations for Scalable RDF Processing. In *Reasoning Web*, volume 6848 of *Lecture Notes in Computer Science*, pages 202–249. Springer, 2011. doi:10.1007/978-3-642-23032-5_4.
- 84 Yu Hou, Jeremy Yeung, Hua Xu, Chang Su, Fei Wang, and Rui Zhang. From answers to insights: Unveiling the strengths and limitations of ChatGPT and biomedical knowledge graphs. *medRxiv*, jun 2023.
- 85 Toshihide Ibaraki and Tiko Kameda. On the optimal nesting order for computing n-relational joins. *ACM Trans. Database Syst.*, 9(3):482–502, 1984. doi:10.1145/1270.1498.
- 86 Dilshod Ibragimov, Katja Hose, Torben Bach Pedersen, and Esteban Zimányi. Processing Aggregate Queries in a Federation of SPARQL Endpoints. In *ESWC*, volume 9088 of *Lecture Notes in Computer Science*, pages 269–285. Springer, 2015. doi:10.1007/978-3-319-18818-8_17.

- 87 Enrique Iglesias, Samaneh Jozashoori, David Chaves-Fraga, Diego Collarana, and Maria-Esther Vidal. Sdm-rdfizer: An RML interpreter for the efficient creation of RDF knowledge graphs. In Mathieu d'Aquin, Stefan Dietze, Claudia Hauff, Edward Curry, and Philippe Cudré-Mauroux, editors, *CIKM '20: The 29th ACM International Conference on Information and Knowledge Management, Virtual Event, Ireland, October 19-23, 2020*, pages 3039–3046. ACM, 2020. doi:10.1145/3340531.3412881.
- 88 Maciej Janik, Ansgar Scherp, and Steffen Staab. The Semantic Web: Collective Intelligence on the Web. *Informatik Spektrum*, 34(5):469–483, 2011. doi:10.1007/s00287-011-0535-x.
- 89 Simon Jupp, Sean Bechhofer, and Robert Stevens. SKOS with OWL: don't be full-ish! In Catherine Dolbear, Alan Ruttenberg, and Ulrike Sattler, editors, *Proceedings of the Fifth OWLED Workshop on OWL: Experiences and Directions, collocated with the 7th International Semantic Web Conference (ISWC-2008), Karlsruhe, Germany, October 26-27, 2008*, volume 432 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2008. URL: https://ceur-ws.org/Vol-432/owled2008eu_submission_22.pdf.
- 90 Pallika Kanani, Andrew McCallum, and Chris Pal. Improving author coreference by resource-bounded information gathering from the web. In *Conference on Artificial Intelligence*. Morgan Kaufmann, 2007. URL: <http://ijcai.org/Proceedings/07/Papers/067.pdf>.
- 91 Gjergji Kasneci, Shady Elbassuoni, and Gerhard Weikum. MING: Mining Informative Entity Relationship Subgraphs. In *Information and Knowledge Management*. ACM, 2009. doi:10.1145/1645953.1646196.
- 92 Andreas Kasten. *Secure semantic web data management: confidentiality, integrity, and compliant availability in open and distributed networks*. PhD thesis, University of Koblenz and Landau, Germany, 2016. URL: <https://kola.opus.hbz-nrw.de/frontdoor/index/index/docId/1393>.
- 93 Andreas Kasten, Ansgar Scherp, and Peter Schaub. A framework for iterative signing of graph data on the web. In *Extended Semantic Web Conference*. Springer, 2014. doi:10.1007/978-3-319-07443-6_11.
- 94 Ankesh Khandelwal, Ian Jacobi, and Lalana Kagal. Linked rules: Principles for rule reuse on the web. In Sebastian Rudolph and Claudio Gutierrez, editors, *Web Reasoning and Rule Systems - 5th International Conference, RR 2011, Galway, Ireland, August 29-30, 2011. Proceedings*, volume 6902 of *Lecture Notes in Computer Science*, pages 108–123. Springer, 2011. doi:10.1007/978-3-642-23580-1_9.
- 95 Holger Knublauch and Dimitris Kontokostas. Shapes constraint language (shacl). W3C Recommendation, 2017. URL: <https://www.w3.org/TR/2017/REC-shacl-20170720/>.
- 96 Christoph Lange, Jörg Langkau, and Sebastian R. Bader. The IDS information model: A semantic vocabulary for sovereign data exchange. In Boris Otto, Michael ten Hoppel, and Stefan Wrobel, editors, *Designing Data Spaces: The Ecosystem Approach to Competitive Advantage*, pages 111–127. Springer, 2022. doi:10.1007/978-3-030-93975-5_7.
- 97 Georg Lausen, Michael Meier, and Michael Schmidt. Sparql constraints for RDF. In Alfons Kemper, Patrick Valduriez, Nouredine Mouadib, Jens Teubner, Mokrane Bouzeghoub, Volker Markl, Laurent Amsaleg, and Ioana Manolescu, editors, *EDBT 2008, 11th International Conference on Extending Database Technology, Nantes, France, March 25-29, 2008, Proceedings*, volume 261 of *ACM International Conference Proceeding Series*, pages 499–509. ACM, 2008. doi:10.1145/1353343.1353404.
- 98 Maurizio Lenzerini. Data integration: A theoretical perspective. In *Proceedings of the Twenty-first ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 3-5, Madison, Wisconsin, USA*, pages 233–246. ACM, 2002. doi:10.1145/543613.543644.
- 99 Alberto Moya Loustaunau and Aidan Hogan. Predicting SPARQL query dynamics. In *K-CAP*, pages 161–168. ACM, 2021. doi:10.1145/3460210.3493565.
- 100 Mohamed Nadjib Mami, Damien Graux, Simon Scerri, Hajira Jabeen, Sören Auer, and Jens Lehmann. Squerall: Virtual ontology-based access to heterogeneous and large data sources. In *International Semantic Web Conference*, pages 229–245. Springer, 2019. doi:10.1007/978-3-030-30796-7_15.
- 101 Deborah L. McGuinness. Ontologies come of age. In Dieter Fensel, James A. Hendler, Henry Lieberman, and Wolfgang Wahlster, editors, *Spinning the Semantic Web: Bringing the World Wide Web to Its Full Potential [outcome of a Dagstuhl seminar]*, pages 171–194. MIT Press, 2003.
- 102 Gabriela Montoya, Hala Skaf-Molli, and Katja Hose. The Odyssey Approach for Optimizing Federated SPARQL Queries. In *ISWC*, volume 10587 of *Lecture Notes in Computer Science*, pages 471–489. Springer, 2017. doi:10.1007/978-3-319-68288-4_28.
- 103 Boris Motik, Ian Horrocks, and Ulrike Sattler. Adding integrity constraints to owl. In *OWLED 2007 - OWL: Experiences and Directions*, volume 258, Aachen, 2007. CEUR Workshop Proceedings (CEUR-WS.org). URL: <https://ceur-ws.org/Vol-258/paper11.pdf>.
- 104 Boris Motik, Ian Horrocks, and Ulrike Sattler. Bridging the gap between owl and relational databases. *Web Semantics: Science, Services and Agents on the World Wide Web*, 7(2):74–89, 2009. doi:10.1016/j.websem.2009.02.001.
- 105 Boris Motik, Ulrike Sattler, and Rudi Studer. Query answering for OWL-DL with rules. In *International Semantic Web Conference*. Springer, 2004. doi:10.1007/978-3-540-30475-3_38.
- 106 Thomas Neumann and Gerhard Weikum. RDF-3X: a risc-style engine for RDF. *Proc. VLDB Endow.*, 1(1):647–659, 2008. doi:10.14778/1453856.1453927.
- 107 Natalya Fridman Noy, Yuqing Gao, Anshu Jain, Anant Narayanan, Alan Patterson, and Jamie Taylor. Industry-scale knowledge graphs: lessons and

- challenges. *Commun. ACM*, 62(8):36–43, 2019. doi:10.1145/3331166.
- 108 Natasha Noy, Yuqing Gao, Anshu Jain, Anant Narayanan, Alan Patterson, and Jamie Taylor. Industry-scale Knowledge Graphs: Lessons and Challenges. *Communications of the ACM*, 62(8):36–43, 2019. doi:10.1145/3331166.
 - 109 Daniel Oberle. *Semantic Management of Middleware*. Springer, 2006. doi:10.1007/0-387-27631-9.
 - 110 Daniel Oberle, Anupriya Ankolekar, Pascal Hitzler, Philipp Cimiano, Michael Sintek, Malte Kiesel, Babak Mougouie, Stephan Baumann, Shankar Vembu, Massimo Romanelli, Paul Buitelaar, Ralf Engel, Daniel Sonntag, Norbert Reithinger, Berenike Loos, Hans-Peter Zorn, Vanessa Micelli, Robert Porzel, Christian Schmidt, Moritz Weiten, Felix Burkhardt, and Jianshen Zhou. DOLCE ergo SUMO: On foundational and domain models in the SmartWeb Integrated Ontology (SWIntO). *Web Semant.*, 5(3):156–174, sep 2007. doi:10.1016/j.websem.2007.06.002.
 - 111 Daniel Oberle, Nicola Guarino, and Steffen Staab. What is an ontology? In Steffen Staab and Ruder Studer, editors, *Handbook on Ontologies*. Springer, 2nd edition, 2009. doi:10.1007/978-3-540-92673-3_0.
 - 112 Sitt Min Oo, Gerald Haesendonck, Ben De Meester, and Anastasia Dimou. Rmlstreamer-siso: an rdf stream generator from streaming heterogeneous data. In *International Semantic Web Conference*, pages 697–713. Springer, 2022. doi:10.1007/978-3-031-19433-7_40.
 - 113 Shirui Pan, Linhao Luo, Yufei Wang, Chen Chen, Jiapu Wang, and Xindong Wu. Unifying large language models and knowledge graphs: A roadmap. *CoRR*, abs/2306.08302, 2023. doi:10.48550/arXiv.2306.08302.
 - 114 Yannis Papakonstantinou, Hector Garcia-Molina, and Jennifer Widom. Object Exchange Across Heterogeneous Information Sources. In *Data Engineering*, pages 251–260, Washington, DC, USA, 1995. IEEE Computer Society. doi:10.1109/ICDE.1995.380386.
 - 115 Jorge Pérez, Marcelo Arenas, and Claudio Gutierrez. Semantics and complexity of SPARQL. *ACM Trans. Database Syst.*, 34(3):16:1–16:45, 2009. doi:10.1145/1567274.1567278.
 - 116 Freddy Priyatna, Óscar Corcho, and Juan F. Sequeda. Formalisation and experiences of r2rml-based SPARQL to SQL query translation using morph. In *23rd International World Wide Web Conference, WWW '14, Seoul, Republic of Korea, April 7-11, 2014*, pages 479–490. ACM, 2014. doi:10.1145/2566486.2567981.
 - 117 Eric Prudhommeaux, Iovka Boneva, Jose Emilio Labra Gayo, and Gregg Kellogg. Shex - shape expressions, 2018. URL: <https://shex.io/shex-semantics/index.html>.
 - 118 Kashif Rabbani, Matteo Lissandrini, and Katja Hose. Optimizing SPARQL Queries using Shape Statistics. In *EDBT*, pages 505–510. OpenProceedings.org, 2021. doi:10.5441/002/edbt.2021.59.
 - 119 Kashif Rabbani, Matteo Lissandrini, and Katja Hose. SHACL and ShEx in the Wild: A Community Survey on Validating Shapes Generation and Adoption. In *WWW (Companion Volume)*, pages 260–263. ACM, 2022. doi:10.1145/3487553.3524253.
 - 120 Kashif Rabbani, Matteo Lissandrini, and Katja Hose. Extraction of Validating Shapes from very large Knowledge Graphs. *Proc. VLDB Endow.*, 16(5):1023–1032, 2023. doi:10.14778/3579075.3579078.
 - 121 Kashif Rabbani, Matteo Lissandrini, and Katja Hose. SHACTOR: Improving the Quality of Large-Scale Knowledge Graphs with Validating Shapes. In *SIGMOD Conference Companion*, pages 151–154. ACM, 2023. doi:10.1145/3555041.3589723.
 - 122 Yves Raimond, Christopher Sutton, and Mark B. Sandler. Interlinking Music-Related Data on the Web. *IEEE MultiMedia*, 16(2):52–63, 2009. doi:10.1109/MMUL.2009.29.
 - 123 Steffen Rendle and Lars Schmidt-Thieme. Object identification with constraints. In *International Conference on Data Mining*. IEEE, 2006. doi:10.1109/ICDM.2006.117.
 - 124 Philipp D. Rohde. SHACL constraint validation during SPARQL query processing. In Philip A. Bernstein and Tilmann Rabl, editors, *Proceedings of the VLDB 2021 PhD Workshop co-located with the 47th International Conference on Very Large Databases (VLDB 2021), Copenhagen, Denmark, August 16, 2021*, volume 2971 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2021. URL: <https://ceur-ws.org/Vol-2971/paper05.pdf>.
 - 125 Philipp D. Rohde and Maria-Esther Vidal. Towards certified distributed query processing. In *Joint Proceedings of the ESWC 2023 Workshops and Tutorials co-located with 20th European Semantic Web Conference (ESWC 2023), Heraklion, Greece, May 28-29, 2023*, volume 3443 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2023. URL: https://ceur-ws.org/Vol-3443/ESWC_2023_TrusDeKW_paper_4738.pdf.
 - 126 Carsten Saathoff and Ansgar Scherp. Unlocking the semantics of multimedia presentations in the web with the multimedia metadata ontology. In *International Conference on World Wide Web*. ACM, 2010. doi:10.1145/1772690.1772775.
 - 127 Siddhartha Sahu, Amine Mhedhbi, Semih Salihoglu, Jimmy Lin, and M. Tamer Özsu. The ubiquity of large graphs and surprising challenges of graph processing: extended survey. *VLDB J.*, 29(2-3):595–618, 2020. doi:10.1007/S00778-019-00548-X.
 - 128 Sherif Sakr, Angela Bonifati, Hannes Voigt, Alexandru Iosup, Khaled Ammar, Renzo Angles, Walid G. Aref, Marcelo Arenas, Maciej Besta, Peter A. Boncz, Khuzaima Daudjee, Emanuele Della Valle, Stefania Dumbrava, Olaf Hartig, Bernhard Haslhofer, Tim Hegeman, Jan Hidders, Katja Hose, Adriana Iamnitchi, Vasiliki Kalavri, Hugo Kapp, Wim Martens, M. Tamer Özsu, Eric Peukert, Stefan Plantikow, Mohamed Ragab, Matei Ripeanu, Semih Salihoglu, Christian Schulz, Petra Selmer, Juan F. Sequeda, Joshua Shinavir, Gábor Szárnyas, Riccardo Tommasini, Antonino Tumeo, Alexandru Uta, Ana Lucia Varbanescu, Hsiang-Yun Wu, Nikolay Yakovets, Da Yan, and Eiko Yoneki. The future is big graphs: a commu-

- nity view on graph processing systems. *Commun. ACM*, 64(9):62–71, 2021. doi:10.1145/3434642.
- 129 Ansgar Scherp, Thomas Franz, Carsten Saathoff, and Steffen Staab. A core ontology on events for representing occurrences in the real world. *Multimedia Tools Appl.*, 58(2):293–331, 2012. doi:10.1007/s11042-010-0667-z.
- 130 Ansgar Scherp and Gerd Gröner. Semantic web. In Günther Görz, Ute Schmid, and Tanya Braun, editors, *Handbuch der Künstlichen Intelligenz*, 6. Auflage, pages 783–816. De Gruyter, 2020. doi:10.1515/9783110659948-018.
- 131 Ansgar Scherp, Carsten Saathoff, Thomas Franz, and Steffen Staab. Designing core ontologies. *Applied Ontology*, 6(3):177–221, 2011. doi:10.3233/AO-2011-0096.
- 132 Wolfgang Scheufele and Guido Moerkotte. On the complexity of generating optimal plans with cross products. In Alberto O. Mendelzon and Z. Meral Özsoyoglu, editors, *Proceedings of the Sixteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, 1997. doi:10.1145/263661.263687.
- 133 Falko Schönteich, Andreas Kasten, and Ansgar Scherp. A pattern-based core ontology for product lifecycle management based on DUL. In *Workshop on Ontology Design and Patterns*, volume 2195 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2018. URL: <http://ceur-ws.org/Vol-2195>.
- 134 Andreas Schwarte, Peter Haase, Katja Hose, Ralf Schenkel, and Michael Schmidt. Fedx: Optimization techniques for federated query processing on linked data. In *The Semantic Web - ISWC 2011 - 10th International Semantic Web Conference, Bonn, Germany, October 23-27, 2011, Proceedings, Part I*, volume 7031 of *Lecture Notes in Computer Science*, pages 601–616. Springer, 2011. doi:10.1007/978-3-642-25073-6_38.
- 135 Juan F. Sequeda and Daniel P. Miranker. Ultrawrap: SPARQL execution on relational data. *J. Web Semant.*, 22:19–39, 2013. doi:10.1016/j.websem.2013.08.002.
- 136 Umutcan Şimşek, Elias Kärle, and Dieter Fensel. RocketRML-A NodeJS implementation of a use-case specific RML mapper. In *Proceeding of the First International Workshop on Knowledge Graph Building*, 2019. URL: <https://ceur-ws.org/Vol-2489/paper5.pdf>.
- 137 Manu Sporny, Amy Guy, Markus Sabadello, Drummond Reed, Manu Sporny, Dave Longley, Markus Sabadello, Drummond Reed, Ori Steele, and Christopher Allen. Decentralized identifiers (DIDs) v1.0: Core architecture, data model, and representations. <https://www.w3.org/TR/did-core/>, 2012.
- 138 Markus Stocker, Andy Seaborne, Abraham Bernstein, Christoph Kiefer, and Dave Reynolds. SPARQL basic graph pattern optimization using selectivity estimation. In Jinpeng Huai, Robin Chen, Hsiao-Wuen Hon, Yunhao Liu, Wei-Ying Ma, Andrew Tomkins, and Xiaodong Zhang, editors, *Proceedings of the 17th International Conference on World Wide Web, WWW 2008, Beijing, China, April 21-25, 2008*, pages 595–604. ACM, 2008. doi:10.1145/1367497.1367578.
- 139 Giorgos Stoilos, Giorgos B. Stamou, Jeff Z. Pan, Vassilis Tzouvaras, and Ian Horrocks. Reasoning with Very Expressive Fuzzy Description Logics. *J. Artif. Intell. Res.*, 30:273–320, 2007. doi:10.1613/jair.2279.
- 140 Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: a core of semantic knowledge. In *International Conference on World Wide Web*. ACM, 2007. doi:10.1145/1242572.1242667.
- 141 Kai Sun, Yifan Ethan Xu, Hanwen Zha, Yue Liu, and Xin Luna Dong. Head-to-tail: How knowledgeable are large language models (llm)? A.K.A. will llms replace knowledge graphs? *CoRR*, abs/2308.10168, 2023. doi:10.48550/arXiv.2308.10168.
- 142 York Sure, Michael Erdmann, Juergen Angele, Steffen Staab, Rudi Studer, and Dirk Wenke. On-toEdit: Collaborative Ontology Development for the Semantic Web. In *International Semantic Web Conference*. Springer, 2002. doi:10.1007/3-540-48005-6_18.
- 143 Thomas Pellissier Tanon, Denny Vrandečić, Sebastian Schaffert, Thomas Steiner, and Lydia Pintscher. From freebase to wikidata: The great migration. In *International Conference on World Wide Web*, 2016. doi:10.1145/2872427.2874809.
- 144 Jiao Tao, Evren Sirin, Jie Bao, and Deborah L. McGuinness. Integrity constraints in OWL. In Maria Fox and David Poole, editors, *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2010, Atlanta, Georgia, USA, July 11-15, 2010*, pages 1443–1448. AAAI Press, 2010. doi:10.1609/aaai.v24i1.7525.
- 145 Milena Trajanoska, Riste Stojanov, and Dimitar Trajanov. Enhancing knowledge graph construction using large language models. *CoRR*, abs/2305.04676, 2023. doi:10.48550/arXiv.2305.04676.
- 146 Despoina Trivela, Giorgos Stoilos, Alexandros Chortaras, and Giorgos B. Stamou. Optimising resolution-based rewriting algorithms for OWL ontologies. *J. Web Semant.*, 33:30–49, 2015. doi:10.1016/j.websem.2015.02.001.
- 147 Giovanni Tummarello, Richard Cyganiak, Michele Catasta, Szymon Danielczyk, and Stefan Decker. Sig.ma: live views on the Web of Data. In *Semantic Web Challenge 2009 at the 8th International Semantic Web Conference (ISWC2009)*, 2009. doi:10.1145/1772690.1772907.
- 148 Michael Uschold. Ontology and database schema: What’s the difference? *Appl. Ontology*, 10(3-4):243–258, 2015. doi:10.3233/AO-150158.
- 149 Michael Uschold and Michael Grüninger. Ontologies and semantics for seamless connectivity. *SIGMOD Rec.*, 33(4):58–64, 2004. doi:10.1145/1041410.1041420.
- 150 Ruben Verborgh, Olaf Hartig, Ben De Meester, Gerald Haesendonck, Laurens De Vocht, Miel Vander Sande, Richard Cyganiak, Pieter Colpaert, Erik Mannens, and Rik Van de Walle. Low-cost queryable linked data through triple pattern fragments. In Matthew Horridge, Marco Rospocher, and Jacco van Ossenberg, editors, *Proceedings of the ISWC 2014 Posters & Demonstrations Track a track within the 13th International Semantic Web Conference, ISWC*

- 2014, Riva del Garda, Italy, October 21, 2014, volume 1272 of *CEUR Workshop Proceedings*, pages 13–16. CEUR-WS.org, 2014. URL: https://ceur-ws.org/Vol-1272/paper_10.pdf.
- 151 Ruben Verborgh, Miel Vander Sande, Olaf Hartig, Joachim Van Herwegen, Laurens De Vocht, Ben De Meester, Gerald Haesendonck, and Pieter Colpaert. Triple pattern fragments: A low-cost knowledge graph interface for the web. *J. Web Semant.*, 37-38:184–206, 2016. doi:10.1016/j.websem.2016.03.003.
 - 152 Maria-Esther Vidal, Simón Castillo, Maribel Acosta, Gabriela Montoya, and Guillermo Palma. On the selection of SPARQL endpoints to efficiently execute federated SPARQL queries. *Trans. Large Scale Data Knowl. Centered Syst.*, 25:109–149, 2016. doi:10.1007/978-3-662-49534-6_4.
 - 153 Maria-Esther Vidal, Edna Ruckhaus, Tomas Lampo, Amadís Martínez, Javier Sierra, and Axel Polleres. Efficiently joining group patterns in SPARQL queries. In *The Extended Semantic Web Conference, ESWC*, 2010. doi:10.1007/978-3-642-13486-9_16.
 - 154 Denny Vrandečić and Markus Krötzsch. Wikidata: a free collaborative knowledgebase. *Commun. ACM*, 57(10):78–85, 2014. doi:10.1145/2629489.
 - 155 W3C OWL Working Group. OWL 2 Web Ontology Language Document Overview. W3C recommendation, W3C, oct 2009. <http://www.w3.org/TR/owl2-overview/>.
 - 156 Michael L. Wick, Aron Culotta, Khashayar Rohanimanesh, and Andrew McCallum. An entity based model for coreference resolution. In *SIAM International Conference on Data Mining*, pages 365–376, 2009. doi:10.1137/1.9781611972795.32.
 - 157 Michael L. Wick, Khashayar Rohanimanesh, Karl Schultz, and Andrew McCallum. A unified approach for schema matching, coreference and canonicalization. In *International Conference on Knowledge Discovery and Data Mining*. ACM, 2008. doi:10.1145/1401890.1401977.
 - 158 Gio Wiederhold. Mediators in the architecture of future information systems. *IEEE Computer*, 25(3):38–49, 1992. doi:10.1109/2.121508.
 - 159 Kevin Wilkinson, Craig Sayers, Harumi A. Kuno, and Dave Reynolds. Efficient RDF storage and retrieval in Jena2. In *International Workshop on Semantic Web and Databases*, 2003.
 - 160 David Wood. Reliable and persistent identification of linked data elements. In David Wood, editor, *Linking Enterprise Data*, pages 149–173. Springer, 2010. doi:10.1007/978-1-4419-7665-9_8.
 - 161 Guohui Xiao, Dag Hovland, Dimitris Bilidas, Martin Rezk, Martin Giese, and Diego Calvanese. Efficient ontology-based data integration with canonical iris. In *The Semantic Web - 15th International Conference, ESWC 2018, Heraklion, Crete, Greece, June 3-7, 2018, Proceedings*, volume 10843 of *Lecture Notes in Computer Science*, pages 697–713. Springer, 2018. doi:10.1007/978-3-319-93417-4_45.
 - 162 Vladimir Zadorozhny, Louiqa Raschid, Maria-Esther Vidal, Tolga Urhan, and Laura Bright. Efficient evaluation of queries in a mediator for websources. In *Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data, Madison, Wisconsin, USA, June 3-6, 2002*, pages 85–96, 2002. doi:10.1145/564691.564702.
 - 163 Amrapali Zaveri, Anisa Rula, Andrea Maurino, Ricardo Pietrobon, Jens Lehmann, and Sören Auer. Quality assessment for linked data: A survey. *Semantic Web Journal*, 7(1):63–93, mar 2015. doi:10.3233/SW-150175.
 - 164 Yuchen Zhuang, Xiang Chen, Tong Yu, Saayan Mitra, Victor Bursztyn, Ryan A. Rossi, Somdeb Sarkhel, and Chao Zhang. Toolchain*: Efficient action space navigation in large language models with a* search. *CoRR*, abs/2310.13227, 2023. doi:10.48550/ARXIV.2310.13227.

Logics for Conceptual Data Modelling: A Review

Pablo R. Fillottrani  

Universidad Nacional del Sur, Bahía Blanca, Argentina

Comisión de Investigaciones Científicas, Provincia de Buenos Aires, Argentina

C. Maria Keet¹  

Department of Computer Science, University of Cape Town, South Africa

Abstract

Information modelling for databases and object-oriented information systems avails of conceptual data modelling languages such as EER and UML Class Diagrams. Many attempts exist to add logical rigour to them, for various reasons and with disparate strengths. In this paper we aim to provide a

structured overview of the many efforts. We focus on aims, approaches to the formalisation, including key dimensions of choice points, popular logics used, and the main relevant reasoning services. We close with current challenges and research directions.

2012 ACM Subject Classification Information systems → Database design and models; Computing methodologies → Description logics; Software and its engineering → Formal language definitions; Software and its engineering → Unified Modeling Language (UML); Theory of computation → Data modeling

Keywords and phrases Conceptual Data Modelling, EER, UML, Description Logics, OWL

Digital Object Identifier 10.4230/TGDK.2.1.4

Category Survey

Received 2023-09-14 **Accepted** 2024-02-08 **Published** 2024-05-03

Editors Aidan Hogan, Ian Horrocks, Andreas Hotho, and Lalana Kagal

Special Issue Trends in Graph Data and Knowledge – Part 2

1 Introduction

Information modelling or conceptual modelling plays an essential role in computing by providing a structured and abstract representation of complex data that sustains each software system. It serves as a foundational step in the development lifecycle, facilitating communication and understanding among stakeholders from the identification of requirements to maintenance. In addition, it promotes a shared vision and a common understanding of the system’s domain information that facilitates interoperability with other systems in unforeseen ways at development time. The latest curriculum recommendations² include conceptual modelling in the undergraduate curriculum and pertinent dimensions are listed with multiple terms in the ACM classification codes, notably, among others: Information management, Data Modeling, Model development and analysis, Enterprise modeling, Entity relationship model, and Unified Modeling Language (UML).

Storey et al. recently described conceptual modelling as “*an activity that occurs during information systems development and use that involves capturing, abstracting, and representing relevant aspects of reality, to support understanding, communication, design, and decision making.*” Conceptual models are comprised of constructs, such as entities, events, goals, attributes, relationships, roles, and processes, connected by well-defined rules.” (emphasis in original) [116]. Here, we focus specifically on *conceptual data modelling* (and thus excluding process and goal modelling)

¹ Corresponding author

² Accessible at <https://www.acm.org/education/curricula-recommendations>



and within that, what has been called *structural conceptual data models* [56] (and thus excluding behavioural aspects like UML’s methods). Popular modelling languages over the years include Extended Entity-Relationship (EER) diagrams for database design [36] and the Natural language Information Analysis Method that evolved into Object-Role Modeling (ORM) [59], design for object-oriented programming with Unified Modeling Language’s (UML) Class Diagrams³, and the Semantics of Business Vocabulary and Business Rules [93] that reuses ORM.

Albeit only one of many topics in computing, conceptual data modelling has been investigated widely. This has also resulted in several surveys and scoping reviews on conceptual data modelling broadly but without logics [116], on ontology-driven conceptual modelling aspects from a modelling side [130, 131] rather than logics, on verification topics for UML class diagrams specifically [52, 109], or only for conceptual model-like artefacts as it pertains to reasoning in the context of scalable data management [107]. Those reviews also indicate that conceptual modelling may span (sub)disciplines. Among others, the human aspects of the modelling process may be assumed to be within the scope of information systems and their formal aspects are within the scope of the computing discipline. The latter involves their quality assessment, and algorithms to, among others, convert such specifications into databases and software applications. Conceptual data models are also used in Artificial Intelligence (AI) to drive the design of intelligent information systems, provided they are given a logic-based specification. By being formalised, complex knowledge – entity types, relationships, attributes, and constraints holding over them – can be captured accurately and passed on to a range of computational tasks. Example tasks include using automated reasoners for classification and satisfiability checking and querying data by making use of, e.g., a Description Logics reasoner [12], test data generation (e.g., [110]), optimising query compilation [124], and explainable machine learning processes [83, 117].

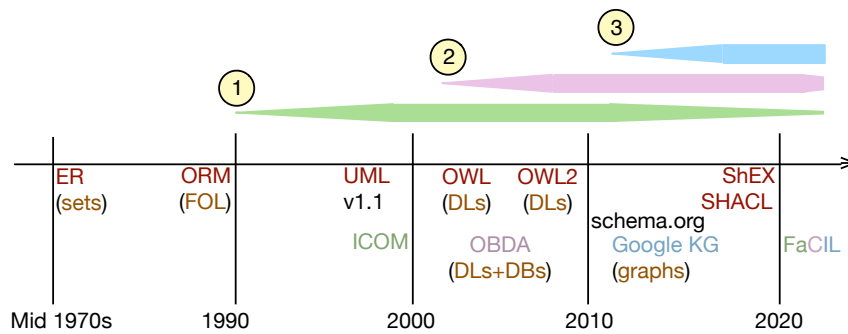
This *logic-based conceptual data modelling*, thus far, has focussed on a number of subtopics, such as which logic to use to formalise the graphical elements and diagram grammar, which features of which conceptual data modelling language to include, and whether one could just have one logic that maps to all major diagram-based conceptual data modelling languages and therewith functioning as a precise interlingua in the back-end – and that for each purpose or assumed application. Figure 1 shows three main strands of investigation and related work programmes with particular aims for logic-based conceptual data modelling together with a few key moments or the publication of pertinent languages and concepts. They are, roughly:

- logic-based reconstructions⁴ of conceptual data models (CDMs⁵) and conceptual data modelling languages (CDMLs) in expressive logics targeting precision and automated reasoning over them, since around 1990.
- runtime usage of CDMs since the early 2000s: ontology-driven information systems including Ontology-Based Data Access (OBDA) where the ontology is de facto a CDM due to being tailored to one application, query optimisation, and verification.
- reach-out to a broader IT scope and to end users since the mid 2010s, which necessarily simplifies and upscales it, where the broader access may have to be tolerant of conflicting information in the model.

³ The first standard listed is version 1.1 from 1997, at <https://www.omg.org/spec/UML/1.1>; last accessed: 30 Sept. 2023.

⁴ The term “reconstruction” captures the process more accurately than “formalisation”. We understand by reconstruction an attempt to get a complete description of information available early at design time, which goes beyond creating just one of many possible formal representations. It includes also, among others, assessment of the graphical design’s implicit assumptions and approach to formalisation. Put differently, the formalisation step forms a part of the reconstruction.

⁵ The abbreviation is well known; tracing it to its origins among the many mentions, it appeared at least already in 1991 [37]. Before that, the CDM abbreviation was also used for Common Data Model or Content Data Model.



■ **Figure 1** Timeline of the three identified strands and a selection of the key moments regarding languages, logics and semantics of the formalisations, and applications. Regarding the latter: ICOM was the first automated reasoner-enabled conceptual modelling tool, Mastro and QuOnto realised OBDA initially, Google’s positioning of KGs helped boost graph-based approaches, and recently FaCIL combines languages and techniques.

Each strand brings with it a different set of requirements for AI theory and techniques, which we will discuss in detail in the paper, and are summarised as follows. The first strand is mostly based on a waterfall design approach: design the model well and shelve it once the system is being implemented. Modellers and domain experts generally develop models only in a graphical language with none or ambiguous semantics that should be formalised. Those logic-based reconstructions focus on formalisations to be as expressive as possible. The more features the better, since the more precision the better, in line with feature extensions from ER to EER [122], ORM to ORM2 [60], and OWL DL to OWL 2DL [38]. In summary, more expressive logics are also more interesting for a broader range of automated reasoning tasks to further help improve a model’s quality.

The second strand of research shifted the focus to leaner languages, designing computationally “well-behaved” fragments of the logics used for the formalisation. The key goal is scalability, not expressiveness, with the logic-based CDM as a component of more advanced AI-driven software systems. It did not focus on reasoning over the CDM itself, but rather the reasoning service as part of querying the data using the conceptual model.

The third, and most recent, strand is ongoing, and might be dubbed “modelling for the masses” and may bifurcate further into new usages. Here, not only scalability is important, but also ease of use and possibly also permitting contradictions, and thus also lower quality models. The latter may happen because the representation language can be too weak to be able to detect quality issues and contradictions. While it may focus on simple queries at most, such basic large models can be of use already in machine learning and natural language processing (NLP) and neuro-symbolic approaches for knowledge graph (KG) embeddings to enhance NLP. Example initiatives that closely relate to CMDs include schema.org, linked data with RDF and optionally with ShEx [13] and SHACL [77] for constraint validation, and the data and modelling component of the community-driven Abstract Wikipedia [132]. This line of work runs in parallel with the first two and may be the most prominent currently.

The different aims of logic-based formalisations, however, do affect how “best” to define that construction, because what “best” entails is relative to the aim. On top of these different aims and tasks, including reasoning tasks, there are various formalisation decisions on how to give semantics to the elements of the diagrammatic notation of the CDM, which, in turn, can affect the computational complexity and therewith the tasks one actually can use the conceptual model for. Owing to the diverse lines of work with their aims for formalising CMDs, different approaches are necessary for assessing a given formalisation in conceptual data modelling. In this condensed

review, using the means of a qualitative narrative review, we zoom in on the evaluation of the aspects that arise in selecting, developing, and applying logic-based semantics in this context. We seek to answer the following questions:

Q1: What are the tasks and challenges in that formalisation?

Q2: Which logics are popular for which (sub-)aim?

Q3: What are the known benefits of a logic-based reconstruction in terms of the outcome and in terms of reasoning services that one may use once a CDM is formalised?

Q4: What are some of the outstanding problems in logic-based conceptual data modelling?

The remainder of the paper is structured as follows. We first describe related work on reviews on conceptual data models in Section 2. Section 3 covers decision points for a formalisation, the logics used for different purposes, and it outlines the two key different processes to do so, covering questions Q1 and Q2. Section 4 identifies and evaluates possible reasoning services that can be applied to CDMs, illustrating with examples two of them. We therewith deal with Q3. Current challenges and future directions related to Q4 are described in Section 5 and we close with conclusions in Section 6.

2 Related Work

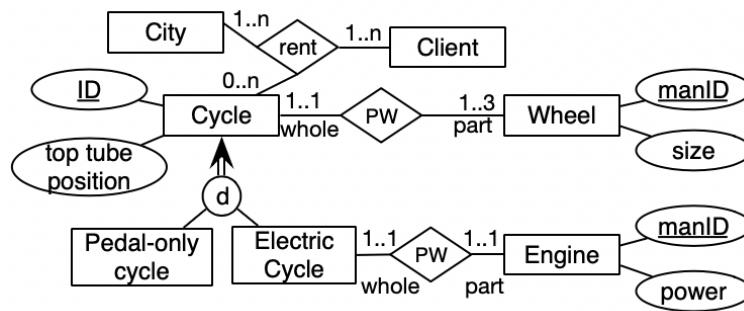
Several reviews of the state of the art in conceptual data modelling and logic-based reconstructions of languages exist, but they either cover only the early years or the first strand of the development of the area [66, 115, 118, 102, 119] or the first and the beginnings of the second strand of the area [2, 39] and are, by now, outdated. New applications of CDMs since those reviews introduce distinctive challenges that were not considered before. Key differences are the uptake of Semantic Web technologies with scalable reasoning over CDMs and runtime usage of CDMs especially for querying data.

Scoping the related work on reviews for CDM to the last 10-15 years, they focus on the non-logical aspects [134, 86, 131, 130, 116]. Wen et al. [134] analysed several quality aspects of conceptual models, such as expressivity, clarity, and semantics, and they evaluated effectiveness of modelling languages in different fields of applications. The formalisation of the languages is generally described, i.e., without logical translations, and no detailed comparison of alternative representations is done.

Other reviews assess CDM from the ontology modelling angle. McDaniel et al. [86] reviewed publications on domain ontology evaluation. Their work concentrates on the evaluation process, and even though domain ontologies are related to conceptual data modelling, the modelling language and their formalisation is not part of the analysed characteristics. Verdonck et al. [131, 130] conducted a systematic literature mapping and review on the domain of ontology-based conceptual modelling. They consider ontology-driven conceptual modelling as the utilisation of ontological techniques, like formal ontology, cognitive science and philosophical logics, to the practice of conceptual modelling. This analyses CDMs in general, not only those approaches related to ontologies.

There are also reviews on the collaborations of the field of conceptual modelling with artificial intelligence [18, 19, 127, 84]. These reviews focus on identifying new research directions and do not address formalisation details.

Gonzalez et al. [52] conducted a systematic literature review of formal verification of structural software models in UML, complemented or not with constraints expressed in textual languages like the Object Constraint Language (OCL). The scope were papers describing research initiatives on model-driven engineering tools that ensure software correctness, and the results classify the type of input models, the reasoning support of the tools, and the completeness of the automatic verification



■ **Figure 2** Example EER diagram of Example 1.

process. The way models are formalised, and how the tools help to develop this formalisation is not analysed, only listing the formal languages used. Shaik et al. [109] presents a more recent literature review with similar aims. They describe language coverage and formalisation techniques in more depth, but the scope is limited to only verifying UML class diagrams, so new applications such as querying are not considered and language coverage is limited to classes, associations, generalisations, compositions, and aggregations. Also, the complexity of formalizations is missing. While such quantitative surveys are useful, they lack in-depth content assessments.

The most recent review, by Storey et al. [116], presents a comprehensive systematic review of the literature in conceptual modelling in general with as aim to identify relevant topics and future research directions. It has a much broader scope including not only static (structural) modelling but also process and collaboration modelling. They recognise the need to support an always increasing variety of users and interconnected domains. Another noteworthy result is that they found out that over the last 15 years, process modelling prevails over data modelling on the research topics. Being a systematic review of a huge amount of literature with semiautomatic tools, there is, however, no reference to logic-based semantic constructions.

There is thus no review on logics for conceptual data modelling specifically, let alone on assessing logic-based formalisations for CDM in view of the current demands and applications not only from the formal point of view, but also on the design decisions that influence data-driven applications across different domains.

3 On formalising conceptual data models

Logic-based reconstructions of CDMs and their languages (CDMLs) used to represent them are motivated by two main key usage scenarios: 1) precision in representation and automated reasoning over them (and, implicitly: quality) and 2) their use at runtime as part of an intelligent information system. It also may be the case that the CDML angle is only a possible scenario and the main aim is to design more logics, whereas from our perspective, the CDMs and CDMLs are the key focus.

This section will summarise the component tasks and types of challenges first, since they set the stage for the logics, subsequently discuss the popular logics used for that, and finally describe the two main approaches typically taken carrying out that task.

3.1 Decision points before the formalisation

This section zooms in on considerations when designing or selecting a logic for creating a logic-based reconstruction of a conceptual data model or modelling language and the decision points involved in it.

In a recent empirical survey, Valle Souza et al. [128] identify six types of functional goals, and five types of quality goals for using conceptual data modelling in practice. Before formalising a CDM, it is important to understand both which subset of the all possible functional goals and which balance of all quality goals are adequate for the context. Different model properties are relevant for achieving these goals, mainly reusability, correctness, comprehensibility, completeness, confinement, and maintainability. Correctness can be further split into precision and coverage.

On the surface, it seems straightforward to formalise CDMs, as something that can be done promptly with little effort, but to get it right for either the whole CDML or an “interesting” fragment requires attention to detail and a substantial amount of knowledge and time. This is due to two key reasons:

- the purpose or reason for the formalisation that influences the design process of a language and therewith the many variations in outcome [46];
- where to set the cut-off point for feature (constraint) inclusion, since if a feature is added, it will be used by someone somewhere and perceived as needed [75].

Purposes such as *reusability*, *comprehensibility*, and *maintainability* favour leaner logics for better performance. In contrast, a purpose of *precision* requires a more expressive logic to maximise coverage of CDML constraints in the ontologically best possible way, which concerns both higher precision so that more unintended models are excluded [55] and philosophical decisions embedded in the logic [47]. Feature inclusion decisions can be split up into two categories. One is modelling features, which concerns whether to include attributes and multi-attribute identifiers, with or without data properties and data types (concrete domains), and which semantics to choose for shared and composite aggregation – among the 23 types of elements and 49 types of constraints across the three main CDML families [76]. The other concerns those that affect the automated reasoning outputs, notably Open World Assumption vs Closed World Assumption and whether to honour the implicit disjointness of classes except when they are in a hierarchy.

In addition to these feature decisions where the logic does not adequately cover all the CDML’s constraints it should be able to express, there is generally a discrepancy in the other direction as well. This concerns the confinement model property, which refers to the degree to which a model has only the necessary information to fulfill its purpose [128]. Here, the logic may permit more than is possible to declare in a diagram due to composition rules of the CDML, with the effect that the logic falls in a higher complexity class than strictly needed.

An overview of the key dimensions of choice points is included in Table 1, which the authors created by combining an assessment of the published logic-based reconstructions (see also Section 3.2) and the top-down approach of the language design procedure introduced in [47]. The first row in Table 1 describes the main aim of the reconstruction, which aligns with the strands 1 vs 2 and 3 introduced in Section 1. The second row presents two approaches to the formalisation: rule based and mapping based. The choice of the approach commits the modeller to a given process, with different tools and outcomes. A detailed analysis for this choice is presented in Section 3.3 and it is further illustrated in the Appendix. Different syntactic and semantic representations for the underlying logic are shown in the third and fourth rows respectively, which summarises the various options that are further discussed in Section 3.2. The next three rows show alternative formalisations for relationship, class disjointness, and how negation is to be treated (closed world view). Finally, the last row describes the influence of the logic-basic constituents in the formalisation, which varies greatly across the published logic-based reconstructions (discussed in Section 3.2.1). The following example illustrates some of these issues, in particular regarding roles and relationships and disjointness.

■ **Table 1** Key dimensions to choose for creating a logic-based reconstruction of a conceptual data model (see Section 3.1 for details).

Dimension	Options	Comments
Main aim	High feature coverage for [precision/automated reasoning], limited features for runtime usage	Mainly a choice between computationally “well-behaved” logic or not
Approach to formalisation	Algorithmic/rules, mappings	See Section 3.3 for details
Syntax	Graphical, textual, both	See Section 3.2 for details
Semantics	Set-based, model-theoretic, graph-based, other	First two are most popular; see Section 3.2 for details
Relationships	See formalisation options in Example 1	Often not stated explicitly which option is chosen
Class disjointness	Classes outside a class hierarchy are disjoint, or not	Most formalisations do <i>not</i> make them disjoint (although assumed in the CDM)
World view	Open, Closed World	Most formalisations are with Open World Assumption
Language feature inclusion	Choose types of elements and constraints to include	A unifying metamodel for EER, UML class diagrams, and ORM2 identified 23 types of elements and 49 constraint types [76] to choose from; see also Section 3.2.1

► **Example 1.** A sample conceptual data model in one of the EER notations is included in Figure 2 (incomplete with respect to the universe of discourse). There are only regular and electric bicycles (which are disjoint) that all have a number of wheels and where the latter has an engine as part. Clients can rent bicycles in a city. Bicycles, engines, and wheels are identified by their respective ID. The first step is to decide how to formalise this in which logic.

Consider the relationship between the **Electric bicycle** and **Engine**. There are multiple options that may have consequences for the logic and the resultant computational complexity of popular automated reasoning tasks:

1. One-directional binary relationship; choose either `partOf` or `hasPart`.
2. Two-directional binary relationships, `partOf` and `hasPart`; choose whether to declare them inverses or not.
3. One non-directional binary relationship with two roles (as part of the relation) that the participating entities play, named, say `partwhole` with as roles `[part]` and `[whole]`; choose whether to define the relationship as having those roles as part.
4. Reify the relationship to a new class and add two new binary relationships to each participating entity type, e.g., `Parthood` with as new relationships `partOf` and `hasPart` that must have as domain `Parthood`; choose whether to approximate the reification (i.e., add only mandatory or only functional (“at most one”) constraints or both on the two new binaries) or demand logical equivalence (i.e., also have the external identifier, as in ORM, or owner entity identifier, as in ER).
5. Acknowledge that relationships in CDMs are local rather than reused like they are in ontologies, so the parthood relationships between **Electric bicycle** and **Engine** and between **Cycle** and **Wheel** must have unique names; choose whether to declare them equivalent or not.

Regarding option 2: adding inverses may or may not change the worst-case computational complexity of a language: e.g., the Description Logic (DL) \mathcal{ALCQ} and \mathcal{ALCQI} are both ExpTime-complete [123], whereas \mathcal{EL} (the basis of the OWL 2 EL profile [90]) does not have inverse properties and is PTime-complete, but adding inverses increases complexity [58].

Option 3 requires more machinery in the logic, specifically roles (called role components in DLs) as core elements and functions to relate the role player to the role, which is used for more convenient processing. Defining relationships, such as defining (familial) aunt to be precisely one's parent's sister ($\text{aunt} \equiv \text{hasParent} \circ \text{hasSister}$), pushes the logic into undecidability in most cases [85, 135, 106]. The reification-based approach of option 4 is used by, e.g., Wikidata's data model⁶. Logical equivalence with a binary relationship requires an advanced identifier constraint that is currently only available in \mathcal{DLR}_{ifd} [31] and $\mathcal{CFDL}_{nc}^{\forall}$ [125] DLs and in full first order logic, but not in OWL that has ample software infrastructure.

For the disjointness, one could either capture that as the complement or as full disjointness, i.e., as $\text{Bicycle} \sqsubseteq \neg \text{Electrical bicycle}$ or as $\text{Bicycle} \sqcap \text{Electrical bicycle} \sqsubseteq \perp$, respectively. Diagrams show disjointness declared on the subsumption relation rather than between the entity types, however, and it thus can be declared only in a class hierarchy, not any number of arbitrary classes in the CDM.

There are more choices for other elements, which, taken together with the myriad of logics, easily can lead to a combinatorial explosion of the combination of formalisation choices with the logic chosen, and which subset of constraints of the CDML is honoured in the formalisation. For instance, OWL DL [87] does not have qualified cardinality constraints to be able to capture the constraint on Wheel fully; OWL 2 DL [91] does. \lrcorner

The different options for this one example are illustrations of how to formalise a particular element, constraint, or combination thereof. CDMs have only a limited set of such patterns and this can be defined algorithmically so that the logic-based reconstruction can be done systematically and a repeat reconstruction will result in the same formalisation, provided the same vocabulary is used where vocabulary needs to be provided. The designers of the different algorithms have made different formalisation choices, and thus their corresponding tools will not necessarily result in the same formalisation given the same CDM.

Finally, an element of the CDML may not be unambiguous and therefore it may be formalised differently across formalisations. The common example of such an issue is UML's aggregation association that was a "semantic variation point" according to the UML v2.4 standard [94] and its semantics is left to the implementer to specify.

3.2 Popular logics for logic-based reconstructions

Most research has focussed on the motivation of the first strand, expressiveness and model quality, both from a conceptual modelling and from a logics perspective, such as [7, 15, 59, 120, 70, 104]. Popular logics to give the graphical elements a formal semantics and to use that for automated reasoning over them at least in theory, are Description Logics (DL) languages but also other logics have been used (e.g., [7, 15]). Several of those other logics, notably those with some tooling support, include UML's object constraint language (OCL) [103], common logic interchange format CLIF (an ISO-standardised first order logic) [98], Alloy (also first-order logic) [21], and Z (a typed first order logic) [67]. Conversely, there are also multiple formalisations for one CDML; e.g., logic-based reconstructions of ORM include, among others, [48, 50, 59, 120, 71, 133] and for ER and EER both from a modeller's perspective [36, 111, 122] and from the logicians' one with the \mathcal{DLR} family [29, 30, 31] and $\mathcal{DL-Lite}$ family [28] of languages.

⁶ Reification such as described by [62]. See also the data model at <https://www.mediawiki.org/wiki/Wikibase/DataModel>; last accessed on 2-1-2024.

■ **Table 2** Popular logics for CDMLs and a set of features (adapted and extended from [42]). “-”: negative/absent; “+”: positive/present; “feature mismatch” refers to the number of constraints (e.g. disjointness) that can be captured in the logic; roles *sensu* DL role components or FOL argument places in relations and relationships.

<i>DL-Lite_A</i> (Approx. OWL 2 QL)	<i>DLR_{iff}</i>	OWL 2 DL	FOL
<i>Selection of features</i>			
- without roles	+ with roles	- without roles	- without roles
- no <i>n</i> -aries	+ has <i>n</i> -aries	- no <i>n</i> -aries	+ has <i>n</i> -aries
+ attributes	+ attributes	+ attributes	- no attributes
+ has datatypes	+ has datatypes	+ has datatypes	- no datatypes
- very few language features; large mismatch	+ little feature mismatch	± some feature mismatch, with overlapping sets	+ little feature mismatch
- logic-based reconstructions to complete	+ logic-based reconstructions exist	- logic-based reconstructions to complete	± logic-based reconstructions exist
+ modularity (import statements etc)	- modularity	+ modularity (import statements etc)	- modularity
UNA / no UNA	no UNA	no UNA	no UNA
± OWA	± OWA	± OWA	± OWA
<i>Computation and implementability</i>			
+ PTIME (TBox); AC ⁰ (ABox)	± ExpTime-complete	± N2ExpTime-complete	- undecidable
+ very scalable (TBox and ABox)	± somewhat scalable (TBox)	± somewhat scalable (TBox)	- not scalable
+ relevant automated reasoners available	- no implementation	+ relevant automated reasoners available	± limited automated reasoners (see text for detail)
+ linking with ontologies doable	- no interoperability	+ linking with ontologies doable	- no interoperability with widely used infrastructures
+ modularity infrastructure	- modularity infrastructure	+ modularity infrastructure	- modularity infrastructure

Alternative approaches consider the verification problem, for which constraint programming is used [25, 26], and there are a few graph-based approaches [20]. Also, there is the deductive databases approach based on logic programming [88], in which the concepts of closed world assumption (CWA) and unique name assumption were first introduced. ConceptBase⁷ is a tool that adds conceptual modelling and metamodelling features based on the same logical representation. Deductive databases focus on a logic-based representation and inference within an already deployed database system, however, where all choices and decisions about the formalisation are already made, while conceptual modelling is concerned with creating a high-level, technology-independent representation of the entire information system during the early stages of development where there are still plenty of open points to formalise. Although it is possible to do conceptual modelling in this context, it is not in the main interest of the deductive databases area. Other attempts, such as exploring category theory [120] for a precise specification, are also considered out of scope for this review.

The next three paragraphs elaborate on the main trends.

⁷ <https://conceptbase.sourceforge.net>

3.2.1 Coverage and DLs

We shall focus on DLs since they are relatively popular thanks to the OWL standard [87, 91], which is largely based on them [65]⁸, the software tooling ecosystem that it fostered, more research has been carried out on logic-based reconstructions into a DL or a DL-based OWL species than for other logic families, and they enjoy ample insights into the computational complexity of language feature combinations.

Most logic-based reconstructions consider only one CDML family at a time. Well-known logics for this purpose are *DL-Lite* and \mathcal{DLR}_{ifd} for EER [7] and UML class diagrams [15], and OWL for (fragments of) ORM and UML class diagrams [133]. The formalisations are typically incomplete with respect to the full CDML due to limited expressiveness of the logic; among others, omitting ER's identifiers (aka keys) [34], excluding n -ary relationships where n may also be ≥ 3 [7, 133], or no special semantics for UML's aggregation association nor for its qualified associations [15]. To some extent, this is unavoidable: ORM and its extended ORM2 are undecidable due to arbitrary projections over n -aries and due to the acyclic role constraint, and probably also due to the antisymmetric role constraint. An advantage of all these formalisations in the different logics covering different features, is that it provides good insight into the computational complexity of the CDMLs. Table 2 lists these and related aspect for four logics, three of which are expressive ones. The for CDMLs relatively well-suited \mathcal{DLR} family – meaning that there is a comparatively good language feature alignment of the logics with CDMLs – are all ExpTime-complete. It varies for the many flavours of *DL-Lite* for different EER fragments [7]. *DL-Lite* is included in the comparison because it is popular in ontology-based data access, where the ontology has to resemble a CDM for seamless query formulation and execution. OWL 2 DL is included for its popularity, given that it is standardised and a reconstruction provides instant access to ample software infrastructure. Their respective language feature sets have some overlap, but either has features that the other one does not have; among others, OWL does not have n -aries proper, no external uniqueness/multi-attribute identifiers or qualified associations, no compound attributes, and no acyclicity, whereas the CDMLs notably do not have property chains and no defined classes. FOL is a common and very expressive language at least on paper and therefore included. Its status of “limited” automated reasoners refers to their plug 'n play level of maturity and the reasoning services they currently offer, as compared to DL-based automated reasoners.

Adding the missing features to any of *DL-Lite*, \mathcal{DLR} or DL-based OWL species is likely to push them straight into undecidability, if they were not already. This also negatively affects obtaining interesting results in unifying the CDMLs through one logic foundation as the central point from which to pivot between graphical CDMs. The typical approach is to identify a common fragment with features that all CDMLs have in common and devise a suitable logic for that, such as the DL \mathcal{ALUNT} [34] and the tailor-made DLs in the same low expressiveness range for evidence-based unification of CDMLs [42]. An exception is the framework for the Distributed Ontology, model, and specification Language (DOL) that uses institutions to provide a framework to let different languages cooperate, including a logic-based reconstruction of UML class diagrams, OWL, and CLIF [89, 54].

Given that one easily arrives at a logic that is ExpTime-complete even without covering all CDML features, little has been done to venture into CDML extensions, although this is also in part because there are not many temporal or spatial conceptual data modelling languages. The main line of research where attempts have been made, concerns expressive temporal DLs like

⁸ OWL DL 2 is based on \mathcal{SROIQ} [64], OWL 2 QL is based on *DL-Lite* [28], OWL 2 EL on \mathcal{EL}^{++} [11], and OWL 2 RL was inspired by both Description Logic Programs [53] and pD* [121].

\mathcal{DLR}_{US} [9] that serves as a basis for the temporal EER ER_{VT} [10]. \mathcal{DLR}_{US} was also explored in context of the MADS spatio-temporal modelling language [99], and ER_{VT} has been extended into EER_{VT}^{++} [95] and TREND [74], all of which still can be reconstructed into \mathcal{DLR}_{US} . While \mathcal{DLR}_{US} turned out to be undecidable [9], this does not need to be the case for all temporal conceptual data models in existence. Only those that have, among others, the following modelling features, are: disjointness and covering (total) constraints, sub-relationships, timestamping, and evolution (i.e., object migration) constraints [6]. Without them, a modeller lacks the ability to represent temporal constraints such as, e.g., “each alumnus must have been a graduating student before”.

3.2.2 CDM runtime usage and DLs

The second strand of research into logic-based reconstructions of CDMLs, runtime usage, focuses on (very) lean fragments for scalability. The software system then uses at least the conceptual model’s vocabulary, relationships, and possibly also its constraints or a subset thereof. Practically, the CDM is then deemed so-called “background knowledge” of the system, rather than the traditional view on it as a starting point for software design from a requirements specification. Popular runtime usages are test data generation for verification and validation [92, 110], query answering with the principal aim of query execution or user-centred query design [16, 33, 35, 82, 113], and database query execution during query compilation [124].

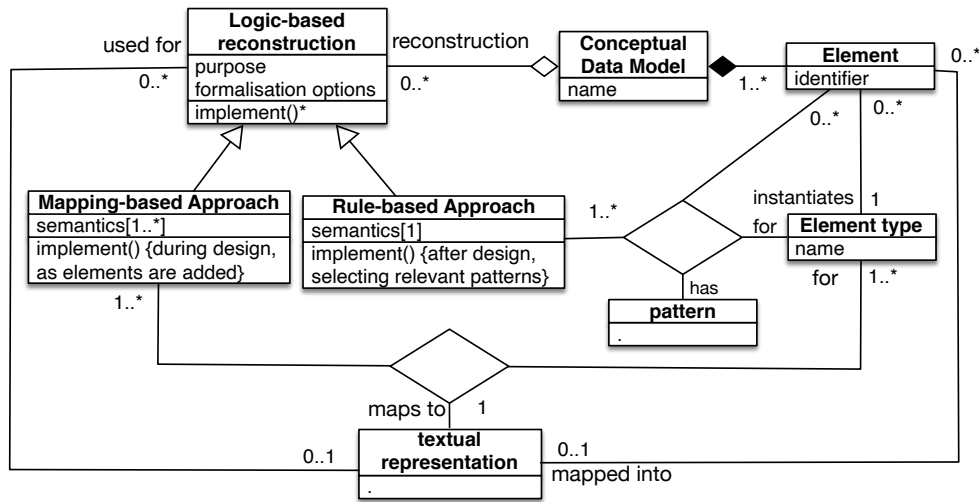
Query answering has received most attention in AI under the name of ontology-based data access (OBDA) [100] and related implementations generally [136, 17, 82, 124, 3], and specific use cases such as EPNet [27] whose “ontologies” are de facto conceptual data models (see for a comparison, e.g., [73]). An alternative approach to the same problem uses transformations rather than a mapping layer, availing of the DL $\mathcal{CFDI}_{nc}^{\forall-}$ and an abstract relational model [125, 105]. $\mathcal{CFDI}_{nc}^{\forall-}$ has been shown to cover a substantial number of constraints used in ORM in its $ORM2_{cd}$ fragment [48] and the approach fits well also with EER [47]. Thanks to the transformations and the assumption of materialising deductions, the expressivity of the logic for the CDML may be higher in this configuration compared to the logic for the CDML in the OBDA approach; other trade-offs are discussed in [47].

For the computationally “well-behaved” lean logics, the key challenge is that the formalisation of a CDM becomes so complicated that it borders cognitive overload for the modeller, if they have to do it all at once. That is, to have to combine in one view and all at the same time the understanding of the universe of discourse, to model it right in the CDM, to know enough of logics, and be fully conversant with its workarounds, convoluted encodings, and approximations. In theory, this should be solvable with good modelling software.

As with the CDM reconstructions that focus on coverage, also here there are steps toward lean temporal fragments, which are motivated mainly by spatio-temporal stream queries with OBDA [69, 40, 97].

3.3 Approaches to the formalisation

Once the CDML, or a fragment thereof, and the logic are chosen there are two main ways to create the logic-based reconstruction, whose components and their interactions are illustrated in Figure 3. The distinction between the two is important, because they meet different sets of formalisation and deployment requirements. One option is to do it algorithmically with a series of rules stating what axiom(s) must be added to the knowledge base for each element encountered in the CDM that needs to be formalised (e.g., [15, 42, 103]). This is like converting an existing informal CDM to the logic. Practically, a particular model is deconstructed into component parts where each component – a pattern or unit for formalisation – may be formalised in a single axiom



■ **Figure 3** Conceptual model describing the characteristics of the two main approaches used for creating logic-based reconstructions of conceptual data models: Mapping-based and rule-based.

or several axioms, depending on the pattern and logic, which are then added one-by-one to a logical theory. This resultant logical theory may be a semantically complete reconstruction of the original CDM or only resembling the original CDM, for it may be missing an element (e.g., a cardinality of “2-4” appears as “ \exists ” in the logical theory) or approximating one (e.g., reification of an n -ary without the identification constraint).

The other option is to declare a new textual syntax of the modelling language, map that syntax to the graphical elements of the CDML, specify the semantics for the syntax, and then show it can be represented in the chosen logic (e.g., [7, 48]). In this second option, the graphical elements in the CDM are effectively a syntactic sugar coating in the modelling process that is already logic-based from the start. With the mapping based approach, it is fully reconstructed by design if the mapping were 1:1 and any excluded features could not be used to begin with, else it is also only an approximation. That is: the details of the reconstruction into the logic vary by proposal and are embedded in the creation of the mapping.

The rules-based approach is illustrated in Appendix A.1, where we adapt the “positionalist core profile” DC_p of [42] for the occasion, which contains the features used most across UML class diagrams, EER, and ORM2, into DL syntax (and thus semantics) [12] with the specific DL role component notation as in the DLR family of DLs [29]. The mapping-based approach is illustrated in Appendix A.2, also with the DC_p language. It is clearly more verbose in its specification than the rules-based one, and takes more time to specify. We illustrate some formalisations with both approaches in the following example.

► **Example 2.** Consider again the bicycles of Figure 2. Let us formalise a section of it into the DL fragment for DC_p , using the rules listed in Appendix A.1:

$$\begin{aligned}
 \geq 1[whole]PW &\sqsubseteq ElectricBicycle \\
 \geq 1[part]PW &\sqsubseteq Engine \\
 Engine &\sqsubseteq \exists power.T \sqcap \leq 1 power \\
 ElectricBicycle &\sqsubseteq Cycle \\
 ElectricBicycle &\sqsubseteq \leq 1[whole]PW \sqcap \geq 1[whole]PW
 \end{aligned}$$

The same section of the model can be formalised a different set of rules for a different logic. For instance, let us take the same section in OWL 2 DL: we first need to somehow add directionality to

the nondirectional PW relationship. Further, one could argue about whether the PW relationship should be typed with a domain and range axiom, since it is used twice and so without typing, one can then obtain a more elegant formalisation. If so, it would be, at least:

```
SubClassOf(ObjectSomeValuesFrom(ex : hasPart) ex : ElectricBicycle)
SubClassOf(ObjectSomeValuesFrom(ex : isPartOf) ex : Engine)
SubClassOf(ex : Engine (ObjectIntersectionOf (DataSomeValuesFrom(ex : power)
  FunctionalDataProperty(ex : power))))
SubClassOf(ex : ElectricBicycle ex : Cycle)
SubClassOf(ex : ElectricBicycle ObjectExactValuesFrom(1ex : hasPart))
```

and optionally with the additional assertion that `hasPart` is the inverse of `isPartOf`. If one were to decide against typing relationships in the rules-based approach, still for OWL 2 DL, then the following set of axioms approximates it by exploiting the qualified cardinality constraint feature:

```
SubClassOf(ex : Engine (ObjectIntersectionOf (DataSomeValuesFrom(ex : power)
  FunctionalDataProperty(ex : power))))
SubClassOf(ex : ElectricBicycle ex : Cycle)
SubClassOf(ex : ElectricBicycle ObjectExactValuesFrom(1 ex : hasPart ex : Engine))
```

The mapping approach, on the other hand, is laborious to define (recall Appendix A.2), but then results in a succinct notation in the formalisation, for one can use the textual version of the CDML. The same model snippet is then:

$$\begin{aligned} \text{REL}(\text{PW}) &= \{whole : \text{ElectricBicycle}, part : \text{Engine}\} \\ \text{ATT}(\text{Engine}) &= \{power : T\} \\ \text{ISA}(\text{ElectricBicycle}, \text{Cycle}) & \\ \text{CMIN}(\text{ElectricBicycle}, \text{PW}, whole) &= 1 \\ \text{CMAX}(\text{ElectricBicycle}, \text{PW}, whole) &= 1 \end{aligned}$$

This notation is likely to be more readable for users who are not logicians, because a term like `ATT` for attribute or, say, `Attribute` in full, is closer to common terminology than `FunctionalDataProperty`, and likewise a simple comma to separate the part and whole versus a “ \sqcap ” symbol. \lrcorner

As can be seen in the example, a different set of rules may result in a knowledge base that is never equivalent, regardless whether that was into the same logic or into different logics with an isomorphism. For instance, with the “bumping up the role names”-approach rather than the roles-based approach, it would not be equivalent due to having created two independent OWL object properties, `hasP` and `isofP`, whereas there is only one relationship (PW) in the \mathcal{DC}_p -based knowledge base. This also motivates that each CDM-to-logic-X converter would need to be explicit on the rules the algorithm uses.

It must be noted that the resultant logic needed to encode all \mathcal{DC}_p knowledge bases, those language features in that DL syntax allow formulas that are *not* \mathcal{DC}_p knowledge bases, or: this logic is more expressive than \mathcal{DC}_p . For example, a knowledge base using that DL fragment may contain $A \sqsubseteq \exists a.T \sqcap \leq 1a \sqcap \forall a.T$, but it cannot be obtained from the translation of any conceptual data model that has only \mathcal{DC}_p 's elements. This is a feature that holds for all such reconstructions: it is a one-way direction from conceptual data model into the logic, but not vice versa.

Observe that since a rule-based construction procedure is linear in the number of elements in the CDM, as most of them are, the overall complexity of translation and any subsequent automated reasoning on the theory remains the same as for the logic. The overall complexity of the mapping-based approach depends on its realisation. If one can model only with what is declared in that mapping, then the complexity is the same as in the logic, which is more efficient

■ **Table 3** Summary of differences between the rule-based and mapping-based approaches, principally emanating from both the different components and relations between them (see Figure 3) and from how the two approaches are currently realised (see the Appendix for an illustration).

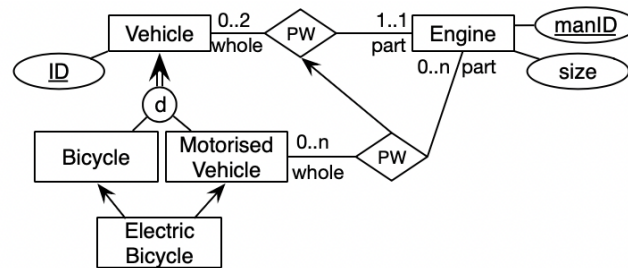
Rule-based	Mapping-based
Logic is more expressive than the CDML	The logic is/can be as expressive as the CDML
When mapped into that logic, the only semantics is that of that logic	Can swap the semantics or declare multiple and choose, like set-based for model-theoretic
Formalisation decisions “hidden” in the algorithm/rules	Ontological commitments explicit in the text-based version and what maps to what
May be with information loss (i.e., less in the formalisation than was modelled in the diagram)	Typically, it is information-preserving
Relatively quick specifications for the formalisation	It is more verbose in its specification and takes more time to specify
Goes in one direction only, from diagram to the axioms	Two-way direction between the CDM and logic
Executed post hoc after completion of the model, or needs to be re-run each time a change has been made	Formalised at modelling time with formalisation and diagram updated in real-time. Computationally faster than re-running the formalisation in the rule-based approach
Graphical elements in the CDM take precedence	Graphical elements in the CDM are effectively a syntactic sugar coating in the modelling process that is already logic-based from the start

than the rules-based approach thanks to not having to do the linear translation. If one can model independently from the CDML in the mapping, then one has to add the pattern-finding complexity to the complexity for the logic.

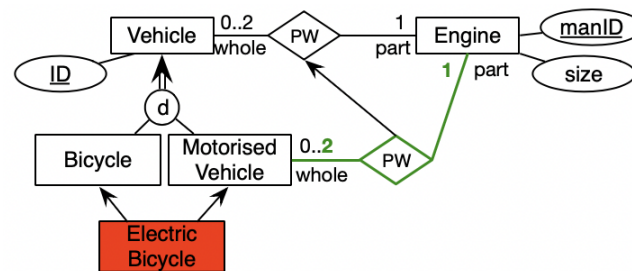
The approaches also can be merged. For instance, a rules-based approach that transforms EER to an intermediary abstract relational model [47], which has its own syntax that is closer to the relational model with its semantics and a mapping from that abstract relational model to a logic (a DL in the case of [17, 82, 124]).

4 Reasoning over and with Conceptual Data Models

Depending on the aims of the modeller, it may already suffice to have a logic-based reconstruction for precision and elimination of ambiguity of the language. It may also be the case that the CDM is formalised in order to use it with automated reasoning services. The principal reasoning tasks assumed for DL-formalised CDMs are the so-called standard reasoning services for DL knowledge bases and OWL ontologies: satisfiability, consistency, instance checking, and querying [12]. Satisfiability and consistency are interesting theoretically, and deducing implicit information can improve on the model’s quality, but for this to be useful during the modelling stage, the available CDML features need to be used more often than currently done [75]. In particular: disjointness constraints, cardinalities beyond 1, and role and relationship subsumption ought, or would need, to be used more often to obtain most benefits. Discovering unsatisfiable classes are useful because if undetected, they result in necessarily empty database tables in a database or OOP classes in the application cannot have any objects. Upfront correction before implementation is better than revising after unit test failures. Detecting implicit cardinality constraints is useful so that they can be made explicit in the database or application, which enhances data integrity. These benefits can be obtained thanks to having formalised the CMD in order to enable automated reasoning over it, which results in a better quality CDM. This is illustrated in the next example.



■ **Figure 4** EER diagram of Example 3, where a modeller created a relationship but forgot to specify it further, and asserted an electrical bicycle to be both a motorised vehicle and a bicycle.



■ **Figure 5** EER diagram of Figure 4 with deductions shown: the Electric bicycle is unsatisfiable (due to the multiple inheritance and disjointness) and the stronger constraints of the parent PW relationship is inherited down the hierarchy.

► **Example 3.** Consider the CDM about bicycles in Figure 4: Vehicles may have at most two engines as part and each engine is part of exactly one vehicle. Bicycles and motorised vehicles (which are disjoint) are vehicles, and electric bicycles are both a type of bicycle and a type of motorised vehicle. In addition, the modeller created a part-whole relationship between motorised vehicle and engine, declared it a subrelationship of the former, but forgot to specify the cardinality constraints, which defaults to 0..n. A logic-based reconstruction of the EER diagram is straightforward in either of the languages in the *DLR* family as well as in OWL 2 and thus also in first order predicate logic, be it following the rules-based or mapping-based approach.

Running the automated reasoner, there are three deductions, which are highlighted in Figure 5. First, the *Electric Bicycle* class is *unsatisfiable*: no individual electric bicycle can be both a bicycle and a motorised vehicle, according to this model, because of the disjointness constraint on the entity type subsumption. Additionally, thanks to the subsumption axiom between the PW between vehicle and engine and motorised vehicle and engine, we obtain two more deductions: the subsumed PW relationship inherits two stronger cardinality constraints declared over the parent relationship. ─

Example 3 is a variation on examples and tooling that exists since 2000 with the ICom tool for EER diagrams [49], its evolution with its own notation⁹ and better module management to declare inter-model assertions [43], and subsequent bifurcation into ORMie for reasoning over ORM2 diagrams [114] and crowd2 that supports reasoning over ORM2, EER, and UML class diagrams and swapping between them [23]. They mostly use the DL *ALCQI* either directly or they use a behind-the-scenes reification of *n*-aries by rewriting them into *n* binaries in case $n \geq 3$. That is, common automated reasoners for OWL 2 DL, such as HermiT [51] or Racer [57], can be, and are, used in these implementations.

⁹ There are other tools that provide a diagrammatic interface to OWL that resemble EER, UML or ORM to a greater or lesser extent, most recently by [79], but the reverse is a different problem and outside the scope of this review, as are graphical notations that are not conceptual data models.

The notion of finite satisfiability – i.e., the problem of deciding whether a knowledge base has a finite non-empty model – in the context of DL-based formalisations is sometimes also considered [14, 15]. If so, this is done more often from the viewpoint of model verification in software engineering and also availing of constraint programming or OCL besides, or instead of, FOL, DL, or HOL, it is focussed on UML class diagrams only, and the majority has only a Yes/No type of output [25, 26, 52, 109].

The reasoning service of instance checking in the DL and OWL sense is not relevant in conceptual model development, for it is focussed on type-level information only, i.e., the TBox in DL terminology. Where instances can, and do, feature in the modelling processes are all in different tasks, being: 1) in the specification of small sample populations to help derive the participation constraints [60], 2) automatic test data generation from CDMs [110], and 3) in a test-driven development method [126]. Because of the absence of an ABox and considering regular conceptual modelling practices, it is hard to obtain an inconsistent CDM and therefore it is, to the best of our knowledge, fully ignored in the research.

Querying over a CDM has not received particular attention, unlike the Query-By-Diagram idea since 1990 [4] and the scalable ontology-based data access that evolved from it [100], which includes using a graphical conceptual data model for it, notably ORM and ORM-like notations (e.g., [33, 35]; see [112] for a review). The first basic task is to use the conceptual data model to “point and click” to select the elements to query, which is then translated into a SPARQL query and from there into SQL, or straight to SQL, to fetch the data from the data store. The more advanced option uses the knowledge represented in the CDM to enhance the query. The enhancement can occur at the level of the TBox, where the query itself is rewritten taking the logic-based reconstruction of the CDM into account, or it is used to compute the deductions over the instances to subsequently materialise the results (i.e., append to the database), which is then queried with the plain query. The general idea is illustrated in Example 2.

► **Example 4.** Consider the following simple conceptual data model CDM that consists of a fragment of the EER diagram in Figure 4:

$$CDM = \{\text{Bicycle} \sqsubseteq \text{Vehicle}, \text{MotorisedVehicle} \sqsubseteq \text{Vehicle}, \text{Bicycle} \sqsubseteq \neg\text{MotorisedVehicle}\}$$

A corresponding database may have three tables, assuming each entity type has its own database table, and at least one instance each:

$$DB = \{\text{Bicycle}(b1), \text{MotorisedVehicle}(mv1), \text{Vehicle}(v1)\}$$

Consider now the query “Retrieve all vehicles”, i.e., `SELECT * FROM Vehicle` for a relational database. A regular RDBMS returns only $\{v1\}$ as answer, it being the only tuple in that table.

Now consider OBDA with the query rewriting approach. The abstract representation of the query is: $q(x) \leftarrow \text{Vehicle}(x)$. In evaluating the query, it first consults CDM : the algorithm detects the two subsumption axioms and rewrites the query as $q(x) \leftarrow \text{Vehicle}(x) \vee \text{Bicycle}(x) \vee \text{MotorisedVehicle}(x)$. This rewritten query is converted into SQL, which amounts to a union of `SELECT * FROM Vehicle`, `SELECT * FROM MotorisedVehicle`, and `SELECT * FROM Bicycle`. It returns $\{b1, mv1, v1\}$, which is what a typical user would expect when asking for all the vehicles.

Consider again the same query, but now we incorporate the knowledge of the CDM in the database *before* we run the same query, also called ABox rewriting or the combined approach. The algorithm detects the $\text{Bicycle} \sqsubseteq \text{Vehicle}$ and updates the database:

$$DB = \{\text{Bicycle}(b1), \text{MotorisedVehicle}(mv1), \text{Vehicle}(v1), \text{Vehicle}(b1)\}$$

and likewise for the motorised vehicle:

$$DB = \{\text{Bicycle}(b1), \text{MotorisedVehicle}(mv1), \text{Vehicle}(v1), \text{Vehicle}(b1), \text{Vehicle}(mv1)\}$$

The query $q(x) \leftarrow \text{Vehicle}(x)$ translates to `SELECT * FROM Vehicle`, but now that table has the other instances as well, and so the query answer is $\{b1, mv1, v1\}$ as well. This approach permits more advanced queries, such as with path queries that have been shown to be more effective [82]. ◻

Notwithstanding that the intuitive idea is seemingly straightforward, the logics and algorithms for the various options are rather involved and depend on the logic used for the TBox; for an early overview on query rewriting see [100], for the first database completion, see [81], and an overview with many further references can be found in Section 3.1 of Schneider and Simkus' recent review on linking ontologies to databases [107]. In addition, each option has its pros and cons regarding computational complexity, query expressiveness, expressivity of the logic for the *CDM*, and optimal usage scenarios [47].

Finally, two orthogonal choices that affect reasoning are choosing between the Closed (CWA) vs Open (OWA) world assumptions and whether to choose for the unique name assumption (UNA) or not. No UNA negatively affects computational complexity especially for the lean OBDA logics [8]. CWA vs OWA principally affects the deductions and it is mostly left implicit that the logic-based reconstructions use OWA since logics in AI assume this unless stated otherwise. CWA is often used in situations where the knowledge is assumed to be complete and where uncertainty is not explicitly represented or tolerated. OWA is used in situations where uncertainty is explicitly acknowledged and where it is important to represent and reason about incomplete information. OWA is more flexible and allows for the representation of unknown or partially known facts. In principle, a CDM may include both approaches.

5 Challenges and future directions

We describe some challenges that emerge from the discussion in previous sections. We classify them along three lines of inquiry: CDM languages, CDM integration with related areas, and CDM applications.

5.1 CDML design

In this section, we analyse challenges in formalisation and expressivity of CDML design. The formalisation space to a logic of choice is crowded with many attempts and assigning semantics to a CDML appears a solved problem, or if not solved, admitted to be intractable in the sense that it will never meet all demands at the same time – good in the formalisation, good in tooling support, and effective use throughout applications. Yet another formalisation of plain EER, ORM2, or UML class diagrams in yet another logic may only make a marginal contribution to the body of knowledge. Also the CDM interoperability, or logic as unifier, task has been well explored (see [42, 22] and references therein), albeit with limited tools and mostly covering a small set of constraints. From our own experiences by both authors, it is tedious, time-consuming, and difficult to publish because from the outside it looks like just more of the same without an appreciation of the thorny finer details and principal differences and consequences thereof. There may be a higher chance of more impact by considering extensions, notably logic-based temporal conceptual data modelling for stream processing or for process mining, and to investigate how to transform a logic-based temporal CDM into a temporal database. This may also connect with the process modelling that Storey et al.'s review highlighted as a general trend in conceptual data modelling [116].

The different clusters of formalisations have different sets of shortcomings and one that they all share, except for the CDML profiles in [42]: none of them is *evidence-based* regarding what features conceptual modellers use and to prioritise accordingly. In addition, user evaluations on usability and understandability are mostly lacking. When carried out, it is about the non-logical additional graphical or textual notation, such as for the logic-based TREND temporal conceptual data modelling language [74] or diagrammatic preferences only (e.g., [129]). Among others, neither the effects of using an automated reasoner in a conceptual data modelling task has

been investigated with human modellers, nor the perceptions of modelling for OBDA. To attain scientific progress, such experiences need to go beyond anecdotes in a few use cases and they need to be tested in a controlled setting, or at least documented and analysed systematically when pooling together a set of use cases.

In addition, to the best of our knowledge, there are no methodologies that incorporate logic-based reconstructions and automated reasoning over the CDM, other than stating it as part of a workflow in FaCiL [22] and it is alluded to in test-driven development of CDM in [126]. Perhaps these gaps contribute to the, to date, limited uptake of logic-based conceptual data modelling in industry. Another reason for that may be that modellers do not use as many language features as they ought to [75] to get the most out of the automated reasoning and thus may need to be trained better in logic-based conceptual data modelling. Another reason may be the relative immaturity of the sparse logic-based conceptual data modelling tools that are mostly of the level of proof-of-concept or prototype [21, 23, 41, 43, 114], rather than full-fledged end-user and commercial-level applications. The underlying issues and opportunities are still unexplored.

Recently, there has been a decline in interest to develop logic-based reasoners [1], most of them being nowadays discontinued in their development. It seems that the mere availability of reasoners is not enough for widespread usage. Much effort is necessary to develop tools that integrate reasoning with real world applications. CDMs may present an opportunity to investigate new tools that demand reasoning services, with the objective to improve the modelling process and enhance the quality of conceptual models, and, just as important, their runtime usage in intelligent information systems.

5.2 CDM integration with related research areas

Ontologies and knowledge graphs are well established research areas that are closely related to CDMs. There are numerous practices that attempt to blur the lines between CDMs and ontologies [45], both in OBDA and elsewhere, as well as recognising the differences but then facing the challenge of how exactly how to relate the two artefacts (e.g., [32, 68]). This involves both how to map between the two at the language level and at the modelling pattern level, and the processes for the various use cases. For instance, top-down generation of candidate CDMs from an ontology, as Jarrar et al.’s aim was [68], requires different procedures from the original motivation for ontologies, as a bottom-up approach to provide a common vocabulary that all CDMs can link to [73]. Their precise interaction – why, when, and how – may be informally clear to some, but that is still non-trivial to apply in practice and does not appear to be clear to the practitioner community. What exactly is missing to fully resolve it both in theory, including with which methods and techniques, and for deployment, is yet to be addressed fully.

This review being a narrative review about the logics rather than a systematic review of both logics and automated reasoning services that are more popular in the ontology engineering field, there is the risk of some bias in the selection of sources. For Section 4, this was intentionally limited to “standard” reasoning services to illustrate some benefits of a logic-based approach. A systematic or broader narrative review about reasoning services for logic-based CDMs may provide additional insight.

From a different angle, and looking at both older and more recent techniques and standards than the most popular – by a large margin in terms of research efforts – logic-based reconstructions, are knowledge graphs that also do relate in some way to CDMs and the logics. Graph-based approaches are expected to (re)gain in popularity. We do not refer to choosing a graph-based semantics after transforming a CDM into OWL or OWL 2 syntax [87, 91], though possible, but the transformation of the CDMs into graphs directly or them being graphs from the outset. The former was proposed by Boyd and McBrien who used hypergraphs for interoperability among the

established CDMLs [20] that is based on their graph-based data model [101]. An example of the latter is the TEGeL modelling language, which is a type graph from the outset and has a set of new icons; its formalisation is only assumed in [108], however, and its aim is specifically for a translation into a graph database only. One could link it to the formal definitions in Appendix B of [63], to RDF [61] or RDFS [24], and with or without constraints, be they with ShEx [13] or SHACL [77] or their logic-based foundation (see [96] for a recent brief overview and references therein), or another approach, such as extending a DL with more options for attributes, as in [78]. This is fertile ground for research whose outcomes may, in turn, feed into the neuro-symbolic strand of usage through KG embeddings that presumably would be improved by such logic-enhanced KGs and it would offer new means for quality control of the information represented in the KG.

5.3 CDM applications in other and new contexts

Other subtopics within the scope of logic-based CDMs are automated content learning, evolution of CDMs, and automated adaptation of CDMs and their database based on the queries posed. Automated CDM content learning should be able to leverage the advances made on corpus-based KG and ontology learning and may also make use of research on automated database-driven logic-based CDM creation that maintains the intermediate state [80]. Dynamic CDM optimisation from database usage patterns concerns updating the CDM by taking into account what data is queried from the database [137]. For instance, if only a fraction of the attributes are queried most of the time, the rarely used ones can be relegated to a separate entity type, like given a *Person* with attributes *tel.no* and *address* where only their *tel.no* is queried in 95% of the queries, then the optimisation suggestion would be to create a separate entity type *Address* with a binary relationship to *Person*. This can save time in query answering and possibly simplify the query interface of an OBDA system.

Further automation in creation, quality control, and use of CDMs does receive interest, as noted in Section 2 and with recent reviews such as [109] on verification. They all do need a formalised CDM to ensure correct operation, yet more such UML advances are still to be ported to other CDMLs and embedded in CDM development methodologies and usage process.

CDM evolution has been well-researched under the term schema evolution at least since the 1990s with renewed interest in the 2000s thanks to ontology evolution and those logics specifically. Ontology evolution is known to be far from trivial, however, which carries over to logic-based CDMs at least to some extent. Given that CDMs have a more restrictive grammar than most of the logics used, it may be less hard, and use cases with current relevance should be specified to limit the possibilities to increase the possibility to find a solution.

Finally, multilingual modelling may become an area of interest, as it is in ontology development for the past 15 years and increasingly for knowledge graphs as well, but it has received little attention in combination with logic-based conceptual modelling [5].

6 Conclusions

Information modelling with conceptual data modelling languages such as EER, UML Class Diagrams, and ORM has been augmented with logic-based reconstructions mainly for precision, quality, and runtime usage for querying and verification. Many logics have been used for many different conceptual data modelling fragments, having used either a rules-based or a mapping-based approach to the formalisation. This paper provided a succinct overview of key choice points in the aims to formalise, approaches to the formalisation, popular logics used, and the principal relevant reasoning services. Current challenges and research directions include the modeller's perspective (with user evaluations), interaction with ontologies, and a renewed interest in graph-based approaches.

References

- 1 Sunitha Abburu. A survey on ontology reasoners and comparison. *International Journal of Computer Applications*, 57(17):33–39, 2012.
- 2 Miguel I. Aguirre-Urreta and George M. Marakas. Comparing conceptual modeling techniques: a critical review of the eer vs. oo empirical literature. *ACM SIGMIS Database: The DATABASE for Advances in Information Systems*, 39(2):9–32, 2008. doi:10.1145/1364636.1364640.
- 3 Lina Al-Jadir, Christine Parent, and Stefano Spaccapietra. Reasoning with large ontologies stored in relational databases: The OntoMinD approach. *Data & Knowledge Engineering*, 69:1158–1180, 2010. doi:10.1016/j.datak.2010.07.006.
- 4 Michele Angelaccio, Tiziana Catarci, and Giuseppe Santucci. QBD*: A graphical query language with recursion. *IEEE Transactions on Software Engineering*, 16(10):1150–1163, 1990. doi:10.1109/32.60295.
- 5 Kutz Arrieta, Pablo R. Fillottrani, and C. Maria Keet. Cosmo: A constructor specification language for abstract wikipedia’s content selection process. Technical report, aug 2023. doi:10.48550/arXiv.2308.02539.
- 6 Alessandro Artale. Reasoning on temporal conceptual schemas with dynamic constraints. In *Proceedings. 11th International Symposium on Temporal Representation and Reasoning, 2004. TIME 2004.*, pages 79–86. IEEE, 2004. doi:10.1109/time.2004.1314423.
- 7 Alessandro Artale, Diego Calvanese, Roman Kontchakov, Vladislav Ryzhikov, and Michael Zakharyashev. Reasoning over extended ER models. In Christine Parent, Klaus-Dieter Schewe, Veda C. Storey, and Bernhard Thalheim, editors, *Proceedings of the 26th International Conference on Conceptual Modeling (ER’07)*, volume 4801 of *LNCS*, pages 277–292. Springer, 2007. Auckland, New Zealand, November 5-9, 2007. doi:10.1007/978-3-540-75563-0_20.
- 8 Alessandro Artale, Diego Calvanese, Roman Kontchakov, and Michael Zakharyashev. DL-Lite without the unique name assumption. In *Proceedings of the 22nd International Workshop on Description Logic (DL’09)*, volume 477 of *CEUR-WS*, 2009. URL: http://ceur-ws.org/Vol1-477/paper_11.pdf.
- 9 Alessandro Artale, Enrico Franconi, Frank Wolter, and Michael Zakharyashev. A temporal description logic for reasoning about conceptual schemas and queries. In S. Flesca, S. Greco, N. Leone, and G. Ianni, editors, *Proceedings of the 8th Joint European Conference on Logics in Artificial Intelligence (JELIA-02)*, volume 2424 of *LNAI*, pages 98–110. Springer Verlag, 2002. doi:10.1007/3-540-45757-7_9.
- 10 Alessandro Artale, Christine Parent, and Stefano Spaccapietra. Evolving objects in temporal information systems. *Annals of Mathematics and Artificial Intelligence*, 50(1-2):5–38, 2007. doi:10.1007/s10472-007-9068-z.
- 11 Franz Baader, Sebastian Brandt, and Carsten Lutz. Pushing the EL envelope. In *Proceedings of the 19th Joint International Conference on Artificial Intelligence (IJCAI’05)*, volume 5, pages 364–369, 2005. URL: <http://ijcai.org/Proceedings/05/Papers/0372.pdf>.
- 12 Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logics Handbook – Theory and Applications*. Cambridge University Press, 2 edition, 2008.
- 13 Thomas Baker and Eric Prud’hommeaux. Shape expressions (shex) 2.1 primer – final community group report. W3C Recommendation, 2019. URL: <http://shex.io/shex-primer/>.
- 14 Mira Balaban and Azzam Maraee. A UML-based method for deciding finite satisfiability in description logics. In Franz Baader, Carsten Lutz, and Boris Motik, editors, *Proceedings of the 21st International Workshop on Description Logics (DL’08)*, volume 353 of *CEUR-WS*, 2008. Dresden, Germany, May 13–16, 2008. URL: <https://ceur-ws.org/Vol1-353/BalabanMaraee.pdf>.
- 15 Daniela Berardi, Diego Calvanese, and Giuseppe De Giacomo. Reasoning on UML class diagrams. *Artificial Intelligence*, 168(1-2):70–118, 2005. doi:10.1016/j.artint.2005.05.003.
- 16 Anthony C. Bloesch and Terry A. Halpin. Conceptual Queries using ConQuer-II. In *Proceedings of ER’97: 16th International Conference on Conceptual Modeling*, volume 1331 of *LNCS*, pages 113–126. Springer, 1997. doi:10.1007/3-540-63699-4_10.
- 17 Alexander Borgida, David Toman, and Grant Weddell. On referring expressions in information systems derived from conceptual modelling. In Isabelle Comyn-Wattiau, Katsumi Tanaka, Il-Yeol Song, Shuichiro Yamamoto, and Motoshi Saeki, editors, *Conceptual Modeling (ER’16)*, volume 9974 of *LNCS*, pages 183–197. Springer, 2016. doi:10.1007/978-3-319-46397-1_14.
- 18 Dominik Bork. Conceptual modeling and artificial intelligence: Challenges and opportunities for enterprise engineering: Keynote presentation at the 11th enterprise engineering working conference (eewc 2021). In *Enterprise Engineering Working Conference*, pages 3–9. Springer, 2021. doi:10.1007/978-3-031-11520-2_1.
- 19 Dominik Bork, Syed Juned Ali, and Ben Roelens. Conceptual modeling and artificial intelligence: A systematic mapping study. Technical report, 2023. doi:10.48550/arXiv.2303.06758.
- 20 Michael Boyd and Peter McBrien. Comparing and transforming between data models via an intermediate hypergraph data model. *Journal on Data Semantics*, IV:69–109, 2005. doi:10.1007/11603412_3.
- 21 Bernardo F. B. Braga, João P. A. Almeida, Giancarlo Guizzardi, and Alessandro Botti Benevides. Transforming OntoUML into Alloy: towards conceptual model validation using a lightweight formal methods. *Innovations in Systems and Software Engineering*, 6(1-2):55–63, 2010. doi:10.1007/s11334-009-0120-5.
- 22 Germán Braun, Pablo R. Fillottrani, and C. Maria Keet. A framework for interoperability between

- models with hybrid tools. *Journal of Intelligent Information Systems*, 60:437–462, 2023. doi:10.1007/s10844-022-00731-7.
- 23 Germán Braun, Giuliano Marinelli, Emiliano Rios Gavagnin, Laura A. Cecchi, and Pablo R. Fillottrani. Web interoperability for ontology development and support with crowd 2.0. In *30th International Joint Conference on Artificial Intelligence, IJCAI'21*, pages 4980–4983, 2021. doi:10.24963/ijcai.2021/707.
 - 24 Dan Brickley and R. V. Guha. Rdf schema 1.1. Standard, W3C, 2014. URL: <https://www.w3.org/TR/2014/REC-rdf-schema-20140225/>.
 - 25 Jordi Cabot, Robert Clarisó, and Daniel Riera. Verification of UML/OCL class diagrams using constraint programming. In *Model Driven Engineering, Verification, and Validation: Integrating Verification and Validation in MDE (MoDeVVA 2008)*, 2008. doi:10.1109/ICSTW.2008.54.
 - 26 Marco Cadoli, Diego Calvanese, Giuseppe De Giacomo, and Toni Mancini. Finite model reasoning on UML class diagrams via constraint programming. In *Proc. of AI*IA 2007*, volume 4733 of *LNAI*, pages 36–47. Springer, 2007. doi:10.1007/978-3-540-74782-6_5.
 - 27 Diego Calvanese, Benjamin Cogrel, Sarah Komla-Ebri, Roman Kontchakov, Davide Lanti, Martin Rezk, Mariano Rodriguez-Muro, and Guohua Xiao. Ontop: Answering SPARQL queries over relational databases. *Semantic Web Journal*, 8(3):471–487, 2017. doi:10.3233/SW-160217.
 - 28 Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Tractable reasoning and efficient query answering in description logics: The DL-Lite family. *Journal of Automated Reasoning*, 39(3):385–429, 2007. doi:10.1007/s10817-007-9078-x.
 - 29 Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. On the decidability of query containment under constraints. In *Proceedings of the 17th ACM SIGACT SIGMOD SIGART Symposium on Principles of Database Systems (PODS'98)*, pages 149–158, 1998. doi:10.1145/275487.275504.
 - 30 Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. Reasoning in expressive description logics with fixpoints based on automata on infinite trees. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI'99)*, pages 84–89, 1999. doi:10.5555/1624218.1624231.
 - 31 Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. Identification constraints and functional dependencies in description logics. In Bernhard Nebel, editor, *Proc. of the 17th Int. Joint Conf. on Artificial Intelligence (IJCAI 2001)*, pages 155–160. Morgan Kaufmann, 2001. Seattle, Washington, USA, August 4-10, 2001. doi:10.5555/1642090.1642111.
 - 32 Diego Calvanese, Tahir Emre Kalayci, Marco Montali, Ario Santoso, and Wil van der Aalst. Conceptual schema transformation in ontology-based data access. In Catherine Faron Zucker, Chiara Ghidini, Amedeo Napoli, and Yannick Toussaint, editors, *Proceedings of the 21st International Conference on Knowledge Engineering and Knowledge Management*, volume 11313 of *LNAI*. Springer, 2018. 12-16 Nov 2018, Nancy, France. doi:10.1007/978-3-030-03667-6_4.
 - 33 Diego Calvanese, C. Maria Keet, Werner Nutt, Mariano Rodríguez-Muro, and Giorgio Stefanoni. Web-based graphical querying of databases through an ontology: the WONDER system. In Sung Y. Shin, Sascha Ossowski, Michael Schumacher, Mathew J. Palakal, and Chih-Cheng Hung, editors, *Proceedings of ACM Symposium on Applied Computing (ACM SAC'10)*, pages 1389–1396. ACM, 2010. March 22-26 2010, Sierre, Switzerland. doi:10.1145/1774088.1774384.
 - 34 Diego Calvanese, Maurizio Lenzerini, and Daniele Nardi. Unifying class-based representation formalisms. *Journal of Artificial Intelligence Research*, 11:199–240, 1999. doi:10.5555/3013545.3013550.
 - 35 Diego Calvanese, Pietro Liuzzo, Alessandro Mosca, José Remesal, Martin Rezk, and Guillem Rull. Ontology-based data integration in epnet: Production and distribution of food during the roman empire. *Engineering Applications of Artificial Intelligence*, 51:212–229, 2016. doi:10.1016/j.engappai.2016.01.005.
 - 36 Peter P. Chen. The entity-relationship model—toward a unified view of data. *ACM Transactions on Database Systems*, 1(1):9–36, 1976. doi:10.1145/320434.320440.
 - 37 HD Crockett, J Guynes, and CW Slinkman. Framework for development of conceptual data modelling techniques. *Information and Software Technology*, 33(2):134–142, 1991. doi:10.1016/0950-5849(91)90058-J.
 - 38 Bernardo Cuenca Grau, Ian Horrocks, Boris Motik, Bijan Parsia, Peter Patel-Schneider, and Ulrike Sattler. OWL 2: The next step for OWL. *Journal of Web Semantics: Science, Services and Agents on the World Wide Web*, 6(4):309–322, 2008. doi:10.1016/j.websem.2008.05.001.
 - 39 Islay Davies, Peter Green, Michael Rosemann, Marta Indulska, and Stan Gallo. How do practitioners use conceptual modeling in practice? *Data & Knowledge Engineering*, 58(3):358–380, 2006. doi:10.1016/j.datak.2005.07.007.
 - 40 Thomas Eiter, Josiane Xavier Parreira, and Patrik Schneider. Spatial ontology-mediated query answering over mobility streams. In E. Blomqvist et al., editors, *Proceedings of the 13th Extended Semantic Web Conference (ESWC'17)*, volume 10249 of *LNCS*, pages 219–237. Springer, 2017. 30 May - 1 June 2017, Portoroz, Slovenia. doi:10.1007/978-3-319-58068-5_14.
 - 41 Carles Farré, Anna Queralt, Guillem Rull, Ernest Teniente, and Toni Urpí. Automated reasoning on UML conceptual schemas with derived information and queries. *Information and Software Technology*, 55(9):1529–1550, 2013. doi:10.1016/j.infsof.2013.02.010.
 - 42 Pablo Fillottrani and C. Maria Keet. Evidence-based lean conceptual data modelling languages. *Journal of Computer Science and Technology*, 21(2):e10, oct 2021. doi:10.24215/16666038.21.e10.
 - 43 Pablo R. Fillottrani, Enrico Franconi, and Sergio Tessaris. The ICOM 3.0 intelligent conceptual modelling tool and methodology. *Semantic*

- Web Journal*, 3(3):293–306, 2012. doi:10.3233/SW-2011-0038.
- 44 Pablo R. Fillottrani and C. Maria Keet. Conceptual model interoperability: a metamodel-driven approach. In A. Bikakis et al., editors, *Proceedings of the 8th International Web Rule Symposium (RuleML'14)*, volume 8620 of *LNCS*, pages 52–66. Springer, 2014. August 18–20, 2014, Prague, Czech Republic. doi:10.1007/978-3-319-09870-8_4.
- 45 Pablo R. Fillottrani and C. Maria Keet. Dimensions affecting representation styles in ontologies. In *1st Iberoamerican conference on Knowledge Graphs and Semantic Web (KGSWC'19)*, volume 1029 of *CCIS*, pages 186–200. Springer, 2019. 24–28 June 2019, Villa Clara, Cuba. doi:10.1007/978-3-030-21395-4_14.
- 46 Pablo R. Fillottrani and C. Maria Keet. An analysis of commitments in ontology language design. In B. Brodaric and F. Neuhaus, editors, *Proceedings of the 11th International Conference on Formal Ontology in Information Systems (FOIS'20)*, volume 330 of *Frontiers in Artificial Intelligence and Applications*, pages 46–60, 2020. doi:10.3233/FAIA200659.
- 47 Pablo R. Fillottrani and C. Maria Keet. KnowID: An architecture for efficient knowledge-driven information and data access. *Data Intelligence*, 2(4):487–512, 2020. doi:10.1162/dint_a_00060.
- 48 Pablo R. Fillottrani, C. Maria Keet, and David Toman. Polynomial encoding of orms conceptual models in $CFDL_{nc}^{\forall}$. In Diego Calvanese and B. Konev, editors, *Proceedings of the 28th International Workshop on Description Logics (DL'15)*, volume 1350 of *CEUR-WS*, pages 401–414, 2015. 7–10 June 2015, Athens, Greece. URL: <https://ceur-ws.org/Vol-1350/paper-50.pdf>.
- 49 E. Franconi and G. Ng. The ICOM tool for intelligent conceptual modelling. In *7th Workshop on Knowledge Representation meets Databases (KRDB'00)*, 2000. Berlin, Germany, 2000. doi:10.5555/2590200.2590206.
- 50 Enrico Franconi, Alessandro Mosca, and Dmitry Solomakhin. The formalisation of ORM2 and its encoding in OWL2. KRDB Research Centre Technical Report KRDB12-2, Faculty of Computer Science, Free University of Bozen-Bolzano, Italy, mar 2012.
- 51 Birte Glimm, Ian Horrocks, Boris Motik, and Giorgos Stoilos. Optimising ontology classification. In *The Semantic Web—ISWC 2010: 9th International Semantic Web Conference, ISWC 2010, Shanghai, China, November 7–11, 2010, Revised Selected Papers, Part I 9*, pages 225–240. Springer, 2010. doi:10.1007/978-3-642-17746-0_15.
- 52 Carlos A. González and Jordi Cabot. Formal verification of static software models in mde: A systematic review. *Information and Software Technology*, 56(8):821–838, 2014. doi:10.1016/j.infsof.2014.03.003.
- 53 Benjamin N. Groszof, Ian Horrocks, Raphael Volz, and Stefan Decker. Description logic programs: Combining logic programs with description logic. In *Proceedings of the 12th International World Wide Web Conference (WWW'03)*, pages 48–57, 2003. Budapest, Hungary. doi:10.1145/775157.775160.
- 54 Object Management Group. Distributed ontology, model, and specification language, feb 2018. URL: <http://www.omg.org/spec/DOL/>.
- 55 Nicola Guarino. Formal ontology and information systems. In N. Guarino, editor, *Proceedings of Formal Ontology in Information Systems (FOIS'98)*, *Frontiers in Artificial Intelligence and Applications*, pages 3–15. Amsterdam: IOS Press, 1998.
- 56 Giancarlo Guizzardi. *Ontological Foundations for Structural Conceptual Models*. Phd thesis, University of Twente, The Netherlands. Telematica Instituut Fundamental Research Series No. 15, 2005.
- 57 Volker Haarslev, Kay Hidde, Ralf Möller, and Michael Wessel. The racerpro knowledge representation and reasoning system. *Semantic Web Journal*, 3(3):267–277, 2012. doi:10.3233/SW-2011-0032.
- 58 Christoph Haase and Carsten Lutz. Complexity of subsumption in the EL family of description logics: Acyclic and cyclic tboxes. In Malik Ghallab, Constantine D. Spyropoulos, Nikos Fakotakis, and Nikolaos M. Avouris, editors, *ECAI 2008 - 18th European Conference on Artificial Intelligence, Patras, Greece, July 21–25, 2008, Proceedings*, volume 178 of *Frontiers in Artificial Intelligence and Applications*, pages 25–29. IOS Press, 2008. doi:10.3233/978-1-58603-891-5-25.
- 59 Terry Halpin. *A logical analysis of information systems: static aspects of the data-oriented perspective*. PhD thesis, University of Queensland, Australia, 1989. URL: https://search.library.uq.edu.au/permalink/f/13gdeh/61UQ_ALMA2179989800003131.
- 60 Terry Halpin and Tony Morgan. *Information modeling and relational databases*. Morgan Kaufmann, 2nd edition, 2008.
- 61 Patrick J. Hayes and Peter F. Patel-Schneider. RDF 1.1 Semantics. W3c recommendation, World Wide Web Consortium, feb 2014. URL: <http://www.w3.org/TR/rdf11-nt/>.
- 62 Daniel Hernández, Aidan Hogan, and Markus Krötzsch. Reifying RDF: what works well with wikidata? In Thorsten Liebig and Achille Fokoue, editors, *Proceedings of the 11th International Workshop on Scalable Semantic Web Knowledge Base Systems 2015*, volume 1457 of *CEUR-WS*, pages 32–47. CEUR-WS.org, 2015. Bethlehem, PA, USA, October 11, 2015. URL: https://ceur-ws.org/Vol-1457/SSWS2015_paper3.pdf.
- 63 Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia d'Amato, Gerard de Melo, Claudio Gutiérrez, José Emilio Labra Gayo, Sabrina Kirrane, Sebastian Neumaier, Axel Polleres, Roberto Navigli, Axel-Cyrille Ngonga Ngomo, Sabbir M. Rashid, Anisa Rula, Lukas Schmelzeisen, Juan F. Sequeda, Steffen Staab, and Antoine Zimmermann. Knowledge graphs. Technical report, 2020. doi:10.48550/arXiv.2003.02320.
- 64 Ian Horrocks, Oliver Kutz, and Ulrike Sattler. The even more irresistible *SROIQ*. In *Proceedings of the Tenth International Conference on Principles of Knowledge Representation and Reasoning (KR'06)*, KR'06, pages 57–67. AAAI Press, 2006. URL: <http://www.aaai.org/Library/KR/2006/kr06-009.php>.

- 65 Ian Horrocks, Peter F. Patel-Schneider, and Frank van Harmelen. From SHIQ and RDF to OWL: The making of a web ontology language. *Journal of Web Semantics*, 1(1):7, 2003. doi:10.1016/j.websem.2003.07.001.
- 66 Richard Hull and Roger King. Semantic database modeling: Survey, applications, and research issues. *ACM Computing Surveys (CSUR)*, 19(3):201–260, 1987. doi:10.1145/45072.45073.
- 67 Amir Jahangard Rafsanjani and Seyed-Hassan Mirian-Hosseiniabadi. A Z Approach to Formalization and Validation of ORM Models. In Ezendu Ariwa and Eyas El-Qawasmeh, editors, *Digital Enterprise and Information Systems*, volume 194 of *CCIS*, pages 513–526. Springer, 2011. doi:10.1007/978-3-642-22603-8_45.
- 68 Mustafa Jarrar, Jan Demey, and Robert Meersman. On using conceptual data modeling for ontology engineering. *Journal on Data Semantics*, 2003. doi:10.1007/978-3-540-39733-5_8.
- 69 Elem Güzel Kalayci, Guohui Xiao, Vladislav Ryzhikov, Tahir Emre Kalayci, and Diego Calvanese. Ontop-temporal: A tool for ontology-based query answering over temporal data. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management, CIKM 2018, Torino, Italy, October 22-26, 2018*, pages 1927–1930, 2018. doi:10.1145/3269206.3269230.
- 70 Ken Kaneiwa and Ken Satoh. Consistency checking algorithms for restricted uml class diagrams. In Jürgen Dix and Stephen J. Hegner, editors, *Foundations of Information and Knowledge Systems*, pages 219–239, Berlin, Heidelberg, 2006. Springer. doi:10.1007/11663881_13.
- 71 C. Maria Keet. Mapping the Object-Role Modeling language ORM2 into Description Logic language \mathcal{DLR}_{ifd} . Technical Report 0702089v2, KRDB Research Centre, Free University of Bozen-Bolzano, Italy, apr 2009. arXiv:cs.LO/0702089v2. doi:10.48550/arXiv.cs/0702089.
- 72 C. Maria Keet. Ontology-driven formal conceptual data modeling for biological data analysis. In Mourad Elloumi and Albert Y. Zomaya, editors, *Biological Knowledge Discovery Handbook: Preprocessing, Mining and Postprocessing of Biological Data*, chapter 6, pages 129–154. Wiley, 2013. doi:10.1002/9781118617151.ch06.
- 73 C. Maria Keet. *An introduction to ontology engineering*, volume 20 of *Computing*. College Publications, UK, 2018. 334p.
- 74 C. Maria Keet and Sonia Berman. Determining the preferred representation of temporal constraints in conceptual models. In H.C. Mayr et al., editors, *36th International Conference on Conceptual Modeling (ER'17)*, volume 10650 of *LNCS*, pages 437–450. Springer, 2017. 6-9 Nov 2017, Valencia, Spain. doi:10.1007/978-3-319-69904-2_33.
- 75 C. Maria Keet and Pablo R. Fillottrani. An analysis and characterisation of publicly available conceptual models. In P. Johannesson, M. L. Lee, S.W. Liddle, A. L. Opdahl, and O. Pastor López, editors, *Proceedings of the 34th International Conference on Conceptual Modeling (ER'15)*, volume 9381 of *LNCS*, pages 585–593. Springer, 2015. 19-22 Oct, Stockholm, Sweden. doi:10.1007/978-3-319-25264-3_45.
- 76 C. Maria Keet and Pablo R. Fillottrani. An ontology-driven unifying metamodel of UML Class Diagrams, EER, and ORM2. *Data & Knowledge Engineering*, 98:30–53, 2015. doi:10.1016/j.datak.2015.07.004.
- 77 Holger Knublauch and Dimitris Kontokostas. Shapes constraint language (shacl). W3C Recommendation, 2017. <https://www.w3.org/TR/shacl/>.
- 78 Markus Krötzsch, Maximilian Marx, Ana Ozaki, and Veronika Thost. Attributed description logics: Reasoning on knowledge graphs. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI'18)*, pages 5309–5313. International Joint Conferences on Artificial Intelligence Organization, jul 2018. doi:10.24963/ijcai.2018/743.
- 79 Domenico Lembo, Valerio Santarelli, Domenico Fabio Savo, and Giuseppe De Giacomo. Graphol: A graphical language for ontology modeling equivalent to owl 2. *Future Internet*, 14(3), 2022. doi:10.3390/fi14030078.
- 80 Lina Lubyte and Sergio Tessaris. Automated extraction of ontologies wrapping relational data sources. In *Proceedings of International Conference on Database and Expert Systems Applications (DEXA'09)*, pages 128–142. Springer, 2009. doi:10.1007/978-3-642-03573-9_10.
- 81 Carsten Lutz, David Toman, and Frank Wolter. Conjunctive query answering in the description logic el using a relational database system. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI'09)*, pages 2070–2075. ACM, 2009. URL: <http://ijcai.org/Proceedings/09/Papers/341.pdf>.
- 82 Weicong Ma, C. Maria Keet, Wayne Oldford, David Toman, and Grant Weddell. The utility of the abstract relational model and attribute paths in sql. In Catherine Faron Zucker, Chiara Ghidini, Amedeo Napoli, and Yannick Toussaint, editors, *Proceedings of the 21st International Conference on Knowledge Engineering and Knowledge Management (EKAW'18)*, volume 11313 of *LNAI*, pages 195–211. Springer, 2018. 12-16 Nov. 2018, Nancy, France. doi:10.1007/978-3-030-03667-6_13.
- 83 Wolfgang Maass, Arturo Castellanos, Monica C. Tremblay, Roman Lukyanenko, and Veda C. Storey. AI explainability: Embedding conceptual models. In Niels Bjørn-Andersen, Roman Beck, Stacie Petter, Tina Blegind Jensen, Tilo Böhmann, Kai-Lung Hui, and Viswanath Venkatesh, editors, *Proceedings of the 43rd International Conference on Information Systems, ICIS 2022, Digitization for the Next Generation, Copenhagen, Denmark, December 9-14, 2022*. Association for Information Systems, 2022. URL: https://aisel.aisnet.org/icis2022/data_analytics/data_analytics/12.
- 84 Wolfgang Maass and Veda C Storey. Pairing conceptual modeling with machine learning. *Data & Knowledge Engineering*, 134:101909, 2021. doi:10.1016/j.datak.2021.101909.
- 85 Fabio Massacci. Decision procedures for expressive description logics with intersection, composition, converse of roles and role identity. In *Proceedings of the 17th International Joint Conference on Ar-*

- tificial Intelligence (IJCAI'2001)*, pages 193–198, 2001.
- 86 Melinda McDaniel and Veda C Storey. Evaluating domain ontologies: clarification, classification, and challenges. *ACM Computing Surveys (CSUR)*, 52(4):1–44, 2019. doi:10.1145/3329124.
 - 87 Deborah L. McGuinness and Frank van Harmelen. OWL Web Ontology Language Overview. W3C Recommendation., 2004. URL: <http://www.w3.org/TR/owl-features/>.
 - 88 Jack Minker. *Foundations of deductive databases and logic programming*. Morgan Kaufmann, 2014.
 - 89 Till Mossakowski, Christoph Lange, and Oliver Kutz. Three semantics for the core of the Distributed Ontology Language. In Michael Grüninger, editor, *Proceedings of the 7th International Conference on Formal Ontology in Information Systems (FOIS'12)*. IOS Press, 2012. 24-27 July, Graz, Austria. doi:10.3233/978-1-61499-084-0-337.
 - 90 Boris Motik, Bernardo Cuenca Grau, Ian Horrocks, Zhe Wu, Achille Fokoue, and Carsten Lutz. OWL 2 Web Ontology Language Profiles. W3C recommendation, W3C, 27 October 2009. URL: <http://www.w3.org/TR/owl2-profiles/>.
 - 91 Boris Motik, Peter F. Patel-Schneider, and Bijan Parsia. OWL 2 web ontology language structural specification and functional-style syntax. W3c recommendation, W3C, 27 October 2009. URL: <http://www.w3.org/TR/owl2-syntax/>.
 - 92 Matthew Nizol, Laura K. Dillon, and R. E. K. Stirewalt. Toward tractable instantiation of conceptual data models using non-semantics-preserving model transformations. In *Proceedings of the 6th International Workshop on Modeling in Software Engineering (MiSE'14)*, pages 13–18. ACM Conference Proceedings, 2014. Hyderabad, India, June 02-03, 2014. doi:10.1145/2593770.2593771.
 - 93 Object Management Group. Semantics of Business Vocabulary and Rules (SBVR) – OMG released versions of SBVR, formal/2008-01-02, jan 2008. URL: <http://www.omg.org/spec/SBVR/1.0>.
 - 94 Object Management Group. Superstructure specification. Standard 2.4.1, Object Management Group, 2012. URL: <http://www.omg.org/spec/UML/2.4.1/>.
 - 95 E. A. N. Ongoma. Formalising temporal attributes in temporal conceptual data models. Msc thesis, Department of Computer Science, 2015.
 - 96 Magdalena Ortiz. A short introduction to SHACL for logicians. In Helle Hvid Hansen, Andre Scedrov, and Ruy J. G. B. de Queiroz, editors, *Proceedings of the 29th International Workshop on Logic, Language, Information, and Computation (WoLLIC'23)*, volume 13923 of LNCS, pages 19–32. Springer, 2023. Halifax, NS, Canada, July 11-14, 2023. doi:10.1007/978-3-031-39784-4_2.
 - 97 Özgür Lütfü Özçep, Ralf Möller, and Christian Neuenstadt. Stream-query compilation with ontologies. In Bernhard Pfahringer and Jochen Renz, editors, *Proceedings of the 28th Australasian Joint Conference on Advances in Artificial Intelligence (AI'15)*, volume 9457 of LNCS, pages 457–463. Springer, 2015. Canberra, ACT, Australia, November 30 – December 4, 2015. doi:10.1007/978-3-319-26350-2_40.
 - 98 Wen-Lin Pan and Da-xin Liu. Mapping object role modeling into common logic interchange format. In *Proceedings of the 3rd International Conference on Advanced Computer Theory and Engineering (IC-ACTE'10)*, volume 2, pages 104–109. IEEE Computer Society, 2010. doi:10.1109/ICACTE.2010.5579141.
 - 99 Christine Parent, Stefano Spaccapietra, and Esteban Zimányi. *Conceptual modeling for traditional and spatio-temporal applications—the MADS approach*. Berlin Heidelberg: Springer Verlag, 2006. doi:10.1007/3-540-30326-X.
 - 100 Antonella Poggi, Domenico Lembo, Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Riccardo Rosati. Linking data to ontologies. *Journal on Data Semantics*, X:133–173, 2008. doi:10.1007/978-3-540-77688-8_5.
 - 101 Alexandra Poulouvassilis and Peter McBrien. A general formal framework for schema transformation. *Data & Knowledge Engineering*, 28(1):47–71, 1998. doi:10.1016/s0169-023x(98)00013-5.
 - 102 Sandeep Puro and Veda C. Storey. A multi-layered ontology for comparing relationship semantics in conceptual models of databases. *Applied Ontology*, 1(1):117–139, 2005. URL: <http://content.iospress.com/articles/applied-ontology/ao000011>.
 - 103 Anna Queralt, Alessandro Artale, Diego Calvanese, and Ernest Teniente. OCL-Lite: Finite reasoning on UML/OCL conceptual schemas. *Data & Knowledge Engineering*, 73:1–22, 2012. doi:10.1016/j.datak.2011.09.004.
 - 104 Anna Queralt and Ernest Teniente. Decidable reasoning in UML schemas with constraints. In Zohra Bellahsene and Michel Léonard, editors, *Proceedings of the 20th International Conference on Advanced Information Systems Engineering (CAiSE'08)*, volume 5074 of LNCS, pages 281–295. Springer, 2008. Montpellier, France, June 16-20, 2008. doi:10.1007/978-3-540-69534-9_23.
 - 105 Jason Saint Jacques, David Toman, and Grant E. Weddell. Object-relational queries over cfdinc knowledge bases: OBDA for the sql-literate. In Subbarao Kambhampati, editor, *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, pages 1258–1264. IJCAI/AAAI Press, 2016. URL: <http://www.ijcai.org/Abstract/16/182>.
 - 106 Manfred Schmidt-Schauß. Subsumption in KL-ONE is undecidable. In *Proceedings of the 1st International Conference on Principles of Knowledge Representation and Reasoning*, pages 421–431, 1989.
 - 107 Thomas Schneider and Mantas Šimkus. Ontologies and data management: A brief survey. *KI-Künstliche Intelligenz*, 34(3):329–353, 2020. doi:10.1007/s13218-020-00686-3.
 - 108 Matthias Sedlmeier and Martin Gogolla. Design and prototypical implementation of an integrated graph-based conceptual data model. In Bernhard Thalheim, Hannu Jaakkola, Yasushi Kiyoki, and Naofumi Yoshida, editors, *Proceedings of the 24th International Conference Information Modelling and Knowledge Bases (EJC'14)*, volume 272 of

- FAIA, pages 376–395. IOS Press, 2014. doi:10.3233/978-1-61499-472-5-376.
- 109 Asadullah Shaikh, Abdul Hafeez, Asif Ali Wagan, Mesfer Alrizq, Abdullah Alghamdi, and Mana Saleh Al Reshan. More than two decades of research on verification of UML class models: A systematic literature review. *IEEE Access*, 9:142461–142474, 2021. doi:10.1109/ACCESS.2021.3121222.
 - 110 Yannis Smaragdakis, Christoph Csallner, and Ranjith Subramanian. Scalable satisfiability checking and test data generation from modeling diagrams. *Automated Software Engineering*, 16:73–99, 2009. doi:10.1007/s10515-008-0044-6.
 - 111 Il-Yeol Song and Peter P. Chen. Entity relationship model. In Ling Liu and M. Tamer Özsu, editors, *Encyclopedia of Database Systems*, volume 1, pages 1003–1009. Springer, 2009. doi:10.1007/978-0-387-39940-9_148.
 - 112 Ahmet Soyly, Martin Giese, Ernesto Jimenez-Ruiz, Evgeny Kharlamov, Dmitriy Zheleznyakov, and Ian Horrocks. Ontology-based end-user visual query formulation: Why, what, who, how, and which? *Universal Access in the Information Society*, 16(2):435–467, jun 2017. doi:10.1007/s10209-016-0465-0.
 - 113 Ahmet Soyly, Evgeny Kharlamov, Dimitry Zheleznyakov, Ernesto Jimenez Ruiz, Martin Giese, Martin G. Skjaeveland, Dag Hovland, Rudolf Schlatte, Sebastian Brandt, Hallstein Lie, and Ian Horrocks. OptiqueVQS: a visual query system over ontologies for industry. *Semantic Web Journal*, 9(5):627–660, 2018. doi:10.3233/sw-180293.
 - 114 Francesco Sportelli and Enrico Franconi. Formalisation of orm derivation rules and their mapping into owl. In *OTM Conferences in Computer Science*, volume 10033, pages 827–843, 2016. doi:10.1007/978-3-319-48472-3_52.
 - 115 Veda C. Storey. Relational database design based on the entity-relationship model. *Data & knowledge engineering*, 7(1):47–83, 1991. doi:10.1016/0169-023X(91)90033-T.
 - 116 Veda C. Storey, Roman Lukyanenko, and Arturo Castellanos. Conceptual modeling: Topics, themes, and technology trends. *ACM Comput. Surv.*, 55(14s), jul 2023. doi:10.1145/3589338.
 - 117 Veda C. Storey, Roman Lukyanenko, Wolfgang Maass, and Jeffrey Parsons. Explainable AI. *Commun. ACM*, 65(4):27–29, 2022. doi:10.1145/3490699.
 - 118 Vijayan Sugumaran and Veda C Storey. Ontologies for conceptual modeling: their creation, use, and management. *Data & knowledge engineering*, 42(3):251–271, 2002. doi:10.1016/S0169-023X(02)00048-4.
 - 119 Vijayan Sugumaran and Veda C Storey. The role of domain ontologies in database design: An ontology management and conceptual modeling environment. *ACM Transactions on Database Systems (TODS)*, 31(3):1064–1094, 2006. doi:10.1145/1166074.1166083.
 - 120 Arthur H. M. ter Hofstede and Henderik A. Proper. How to formalize it? formalization principles for information systems development methods. *Information and Software Technology*, 40(10):519–540, 1998. doi:10.1016/S0950-5849(98)00078-0.
 - 121 Herman J. ter Horst. Completeness, decidability and complexity of entailment for rdf schema and a semantic extension involving the owl vocabulary. *Journal of Web Semantics*, 3(2):79–115, 2005. doi:10.1016/j.websem.2005.06.001.
 - 122 Bernhard Thalheim. Extended entity relationship model. In Ling Liu and M. Tamer Özsu, editors, *Encyclopedia of Database Systems*, volume 1, pages 1083–1091. Springer, 2009. doi:10.1007/978-0-387-39940-9_157.
 - 123 Stephan Tobies. *Complexity Results and Practical Algorithms for Logics in Knowledge Representation*. PhD thesis, RWTH Aachen, 2001. URL: http://sylvester.bth.rwth-aachen.de/dissertationen/2001/082/01_082.pdf.
 - 124 David Toman and Grant E. Weddell. *Fundamentals of Physical Design and Query Compilation*. Synthesis Lectures on Data Management. Morgan & Claypool, 2011. doi:10.1007/978-3-031-01881-7.
 - 125 David Toman and Grant E. Weddell. On adding inverse features to the description logic CFD^V_{nc} . In *PRICAI 2014: Trends in Artificial Intelligence - 13th Pacific Rim International Conference on Artificial Intelligence, Gold Coast, QLD, Australia, December 1-5, 2014.*, pages 587–599, 2014. doi:10.1007/978-3-319-13560-1_47.
 - 126 Albert Tort, Antoni Olivé, and Maria-Ribera Sanchó. An approach to test-driven development of conceptual schemas. *Data & Knowledge Engineering*, 70:1088–1111, 2011. doi:10.1016/j.datak.2011.07.006.
 - 127 Juan Trujillo, Karen C Davis, Xiaoyong Du, Ernesto Damiani, and Veda C. Storey. Conceptual modeling in the era of big data and artificial intelligence: Research topics and introduction to the special issue, 2021. doi:10.1016/j.datak.2021.101911.
 - 128 Isadora Valle Sousa, Tiago Prince Sales, Eduardo Guerra, Luiz Olavo Bonino da Silva Santos, and Giancarlo Guizzardi. What do I get from modeling? an empirical study on using structural conceptual models. In *International Conference on Enterprise Design, Operations, and Computing*, pages 21–38. Springer, 2023. doi:10.1007/978-3-031-46587-1_2.
 - 129 John R. Venable and John C. Grundy. Integrating and supporting entity relationship and object role models. In Mike P. Papazoglou, editor, *Proceedings of the 14th International Conference on Object-Oriented and Entity-Relationship Modeling (ER'95)*, volume 1021 of *LNCIS*, pages 318–328. Springer, 1995. doi:10.1007/BFb0020543.
 - 130 Michaël Verdonck, Frederik Gailly, and Sergio de Cesare. Comprehending 3d and 4d ontology-driven conceptual models: an empirical study. *Information Systems*, 93:101568, 2020. doi:10.1016/j.is.2020.101568.
 - 131 Michaël Verdonck, Frederik Gailly, Sergio De Cesare, and Geert Poels. Ontology-driven conceptual modeling: A systematic literature mapping and review. *Applied Ontology*, 10(3-4):197–227, 2015. doi:10.3233/A0-150154.
 - 132 Denny Vrandečić. Building a multilingual Wikipedia. *Communications of the ACM*, 64(4):38–41, 2021. doi:10.1145/3425778.

- 133 Heba M. Wagih, Doaa S. El Zanfaly, and Mohamed M. Kouta. Mapping Object Role Modeling 2 schemes into $\mathcal{SROIQ}(d)$ description logic. *International Journal of Computer Theory and Engineering*, 5(2):232–237, 2013. doi:10.7763/ijcte.2013.v5.684.
- 134 Kunmei Wen, Yong Zeng, Ruixuan Li, and Jianqiang Lin. Modeling semantic information in engineering applications: a review. *Artificial Intelligence Review*, 37:97–117, 2012. doi:10.1007/s10462-011-9221-2.
- 135 Michael Wessel. Obstacles on the way to qualitative spatial reasoning with description logics: some undecidability results. In Carole A. Goble, Deborah L. McGuinness, Ralf Möller, and Peter F. Patel-Schneider, editors, *Proceedings of the International Workshop in Description Logics (DL'01)*, volume 49 of *CEUR WS*, 2001. Stanford, CA, USA, August 1-3, 2001. URL: <https://ceur-ws.org/Vol-49/Wessel-122start.ps>.
- 136 Guohui Xiao, Linfang Ding, Benjamin Cogrel, and Diego Calvanese. Virtual knowledge graphs: An overview of systems and use cases. *Data Intelligence*, 1:201–223, 2019. doi:10.1162/dint_a_00011.
- 137 Tilmann Zäschke, Stefania Leone, Tobias Gmünder, and Moira C. Norrie. Optimising conceptual data models through profiling in object databases. In Wilfred Ng, Veda C. Storey, and Juan Trujillo, editors, *Proceedings of the 32th International Conference on Conceptual Modeling (ER'13)*, volume 8217 of *LNCIS*, pages 284–297. Springer, 2013. Hong-Kong, November 11-13, 2013. doi:10.1007/978-3-642-41924-9_24.

A Examples of both approaches to the logic-based reconstruction

Since both approaches for CDM or CDML formalisation work in principle for any conceptual data modelling language, as described in Section 3.3, we first harmonise terminology across such CDMLs, using the unified metamodel [76]: Relationship (also called association or fact type), Role (relationship component, association end), Object Type (entity type, class), Attribute¹⁰, and Data Type. Second, the different properties of the rules-based vs. the mapping-based approach, as also depicted in Figure 3, practically result in different sequences of steps for how to create that logic-based reconstruction. The ones we followed in this appendix are graphically depicted in Figure 6, where steps 2 and 3 are illustrated in this appendix; step 1 was taken from other work and steps 4 and 5 are realised in crowd2. Note that the aim is to illustrate the way these logic-based reconstructions are done in the literature and these two approaches are our distillation of the two key distinct approaches researchers have taken. It is not meant to be prescriptive and there is no official or investigated method for carrying out this task.

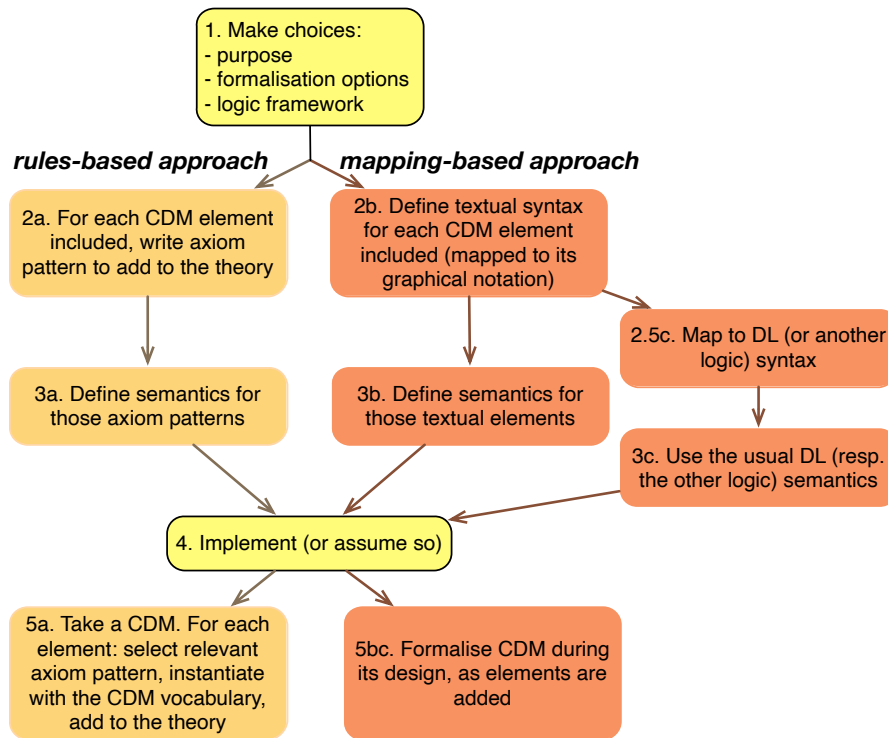
We use the \mathcal{DC}_p “positionalist core profile” of [42] to illustrate both approaches, because it respects the positionalist ontological commitment of those languages and it is a relatively simple language with few features that are those that have been shown to be used most in UML class diagrams, EER, and ORM2. Specifically, based on the analysis of the 101 publicly available CDMs [75], the feature list of \mathcal{DC}_p is expressive enough to include 87.57% of the entities (elements and constraints) used in all the 101 models analysed, and 91,88% of the entities in the UML models, 73.29% of the contents of the ORM and ORM2 models, and 94.64% of the ER and EER models [42].

A.1 Rule-based approach

First, we specify the rules for the algorithmic conversion for any CDM within the \mathcal{DC}_p feature list:

► **Definition 5** (Syntax of \mathcal{DC}_p [42]). *Given a conceptual model in any of UML class diagrams, EER, and ORM2, take the set of all object types ranging over symbols A, B, \dots , binary relationships P , datatypes T and attributes a in the conceptual data model as the basic elements in the knowledge base. Then construct a knowledge base in \mathcal{DC}_p by applying the rules:*

¹⁰ ORM does not have “attribute” as such, but a value type has an attribute, it being a binary relation between a class and a data type. This is a straight-forward conversion procedure; see [44] for details.



■ **Figure 6** Possible sequences of steps for creating logic-based reconstructions of conceptual data models.

1. For each relationship P between object types A and B , add to the knowledge base

$$\geq 1[1]P \sqsubseteq A \text{ and } \geq 1[2]P \sqsubseteq B$$
2. For each attribute a of datatype T within an object type A (including the transformation of ORM's Value Type following the rule in [44]), add

$$A \sqsubseteq \exists a.T \sqcap \leq 1a$$
3. Subsumption between two object types A and B is formalised by adding the assertion

$$A \sqsubseteq B$$
4. For each object type cardinality $m..n$ in relationship P with respect to its i -th component A , add

$$A \sqsubseteq \leq n[i]P \sqcap \geq m[i]P$$
5. Add for each mandatory constraints of a concept A in a relationship P either the axiom

$$A \sqsubseteq \geq 1[1]P \text{ or } A \sqsubseteq \geq 1[2]P$$
 depending on the position played by A in P . This is a special case of the previous one, with $n = 1$.
6. For each single identification in object type A with respect to an attribute a of datatype T , add

$$\text{id } A a$$

Given the formalisation rules in Definition 5, the DL for \mathcal{DC}_p would result in the following syntax: starting from atomic elements, we can construct binary relationships R , arbitrary concepts

4:28 Logics for Conceptual Data Modelling: A Review

C and axioms X as follows:

$$\begin{aligned} C &\longrightarrow \top \mid A \mid \leq k[i]R \mid \geq k[i]R \mid \forall a.T \mid \exists a.T \mid \\ &\leq 1a \mid C \sqcap D \\ R &\longrightarrow \top_2 \mid P \mid (i : C) \\ X &\longrightarrow C \sqsubseteq D \mid \text{id } C a \end{aligned}$$

where $i = 1, 2$ and $0 < k$ (roles may be numbered or named, and are not ordered).

Then, for the second step in the algorithmic procedure, the semantics. For DLs, it can avail of the model-theoretic semantics with its customary notation, as included in Definition 6:

► **Definition 6** (Semantics of \mathcal{DC}_p [42]). *An \mathcal{DC}_p interpretation $\mathcal{I} = (\cdot^{\mathcal{I}}, \cdot^{\mathcal{I}}, \cdot^{\mathcal{I}})$ for a knowledge base in \mathcal{DC}_p consists of a set of objects $\Delta_C^{\mathcal{I}}$, a set of datatype values $\Delta_T^{\mathcal{I}}$, and a function $\cdot^{\mathcal{I}}$ satisfying the constraints shown in Table 4. It is said that \mathcal{I} satisfies the assertion $C \sqsubseteq D$ iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$; and it satisfies the assertion $\text{id } C a$ iff there exists T such that $C^{\mathcal{I}} \subseteq (\exists a.T \sqcap \leq 1a)^{\mathcal{I}}$ (mandatory 1) and for all $v \in T^{\mathcal{I}}$ it holds that $\#\{c \mid c \in \Delta_C^{\mathcal{I}} \wedge (c, v) \in a^{\mathcal{I}}\} \leq 1$ (inverse functional).*

■ **Table 4** Semantics of \mathcal{DC}_p (Source: [42]).

$$\begin{aligned} \top^{\mathcal{I}} &\subseteq \Delta_C^{\mathcal{I}} \\ A^{\mathcal{I}} &\subseteq \top^{\mathcal{I}} \\ \top_2^{\mathcal{I}} &= \top^{\mathcal{I}} \times \top^{\mathcal{I}} \\ P^{\mathcal{I}} &\subseteq \top_2^{\mathcal{I}} \\ T^{\mathcal{I}} &\subseteq \Delta_T^{\mathcal{I}} \\ a^{\mathcal{I}} &\subseteq \top^{\mathcal{I}} \times \Delta_T^{\mathcal{I}} \\ (C \sqcap D)^{\mathcal{I}} &= C^{\mathcal{I}} \cap D^{\mathcal{I}} \\ (\leq k[i]R)^{\mathcal{I}} &= \{c \in \Delta_C^{\mathcal{I}} \mid \#\{(d_1, d_2) \in R^{\mathcal{I}}.d_i = c\} \leq k\} \\ (\geq k[i]R)^{\mathcal{I}} &= \{c \in \Delta_C^{\mathcal{I}} \mid \#\{(d_1, d_2) \in R^{\mathcal{I}}.d_i = c\} \geq k\} \\ (\exists a.T)^{\mathcal{I}} &= \{c \in \Delta_C^{\mathcal{I}} \mid \exists v \in \Delta_T^{\mathcal{I}}.(c, v) \in a^{\mathcal{I}} \wedge v \in T^{\mathcal{I}}\} \\ (\forall a.T)^{\mathcal{I}} &= \{c \in \Delta_C^{\mathcal{I}} \mid \forall v \in \Delta_T^{\mathcal{I}}.(c, v) \in a^{\mathcal{I}} \rightarrow v \in T^{\mathcal{I}}\} \\ (\leq 1a)^{\mathcal{I}} &= \{c \in \Delta_C^{\mathcal{I}} \mid \#\{(c, v) \in a^{\mathcal{I}}\} \leq 1\} \\ (i : C)^{\mathcal{I}} &= \{(d_1, d_2) \in \top_2^{\mathcal{I}} \mid d_i \in C^{\mathcal{I}}\} \end{aligned}$$

An alternative option is to choose either of the five relationship formalisation options described in Example 1, create a conversion algorithm from the positionalist relationships of conceptual data models to that choice, and then create a different profile accordingly with, say, OWL in the formalisation rules. The “bumping up the role names to relationships” choice and DL OWL2 syntax then would add the following axioms to the knowledge base, respectively in the same order as in Definition 5:

1. `SubClassOf(ObjectSomeValuesFrom(ex:hasP) ex:A) and
SubClassOf(ObjectSomeValuesFrom(ex:isOfP) ex:B)`
2. `SubClassOf(ex:A (ObjectIntersectionOf (DataSomeValuesFrom(ex:a)
FunctionalDataProperty(ex:a))))`
3. `SubClassOf(ex:A ex:B)`
4. `SubClassOf(ex:A ObjectIntersectionOf(ObjectMinCardinality(n ex:hasP)
ObjectMaxCardinality(m ex:hasP)))`
5. `SubClassOf(ex:A ObjectSomeValuesFrom(ex:hasP)) or
SubClassOf(ex:A ObjectSomeValuesFrom(ex:isOfP))`

6. `SubClassOf(ex:A DataExactCardinality(1 ex:a))` and
`SubClassOf(ex:A DataSomeValuesFrom(ex:a))`

Alternatively, the relationship is not typed but qualified cardinality constraints are used, or `hasP` and `isOfP` are declared inverses, or only one of the two is introduced with an inverse feature where needed.

An example of this approach with a concrete CDM is illustrated in Example 2.

A.2 Mapping-based approach

The first step in the mapping approach is to declare the textual syntax which, if that were to have been done for \mathcal{DC}_p , would have looked like as in Definition 7. For comprehensiveness, a table with textual elements mapping to the respective graphical elements of the selected modelling language should be done as well (Figure 7), then to declare the semantics (Definition 8). It can stop here, or be mapped into a logic of choice to obtain either a precise or approximate indication of the computational complexity of the language needed for the CDM or chosen fragment thereof, as shown in Definition 9 for a DL. The ones here are based on [72], but includes only those features that are in \mathcal{DC}_p to facilitate a comparison with the first approach.

► **Definition 7** (Conceptual Data Model \mathcal{DC}_p syntax). *A \mathcal{DC}_p conceptual data model is a tuple $\Sigma = (\mathcal{L}, \text{REL}, \text{ATT}, \text{CARD}_R, \text{CARD}_A, \text{ISA}, \text{ID})$ such that:*

1. \mathcal{L} is a finite alphabet partitioned into the sets: \mathcal{C} (class symbols), \mathcal{A} (attribute symbols), \mathcal{R} (relationship symbols), \mathcal{U} (role symbols), and \mathcal{D} (domain symbols); the tuple $(\mathcal{C}, \mathcal{A}, \mathcal{R}, \mathcal{U}, \mathcal{D})$ is the signature of the conceptual model Σ .
2. ATT is a function that maps a class symbol in \mathcal{C} to an \mathcal{A} -labeled tuple over \mathcal{D} , $\text{ATT} : \mathcal{C} \mapsto \mathcal{D}$, so that $\text{ATT}(C) = \{A_1 : D_1, \dots, A_h : D_h\}$ where h a non-negative integer.
3. REL is a function that maps a relationship symbol in \mathcal{R} to an \mathcal{U} -labeled tuple over \mathcal{C} , $\text{REL}(R) = \{U_1 : C_1, U_2 : C_2\}$, if $(U_i, C_i) \in \text{REL}(R)$ (with $i = \{1, 2\}$), then $\text{PLAYER}(R, U_i) = C_i$ and $\text{ROLE}(R, C_i) = U_i$. The signature of the relation is $\sigma_R = \langle \mathcal{U}, \mathcal{C}, \text{PLAYER}, \text{ROLE} \rangle$, where for all $U_i \in \mathcal{U}$, $C_i \in \mathcal{C}$, if $\sharp U \geq \sharp C$ then for each u_i, c_i , $\text{REL}(R)$, we have $\text{PLAYER}(R, U_i) = C_i$ and $\text{ROLE}(R, C_i) = U_i$, and if $\sharp U > \sharp C$ then $\text{PLAYER}(R, U_i) = C_i$, $\text{PLAYER}(R, U_{i+1}) = C_i$ and $\text{ROLE}(R, C_i) = U_i, U_{i+1}$.
4. CARD_R is a function $\text{CARD}_R : \mathcal{C} \times \mathcal{R} \times \mathcal{U} \mapsto \mathbb{N} \times (\mathbb{N} \cup \{\infty\})$ denoting cardinality constraints. We denote with $\text{CMIN}(C, R, U)$ and $\text{CMAX}(C, R, U)$ the first and second component of CARD_R .
5. CARD_A is a function $\text{CARD}_A : \mathcal{C} \times \mathcal{A} \mapsto \mathbb{N} \times (\mathbb{N} \cup \{\infty\})$ denoting multiplicity constraints for attributes. We denote with $\text{CMIN}(C, A)$ and $\text{CMAX}(C, A)$ the first and second component of CARD_A , and $\text{CARD}_A(C, A)$ may be defined only if $(A, D) \in \text{ATT}(C)$ for some $D \in \mathcal{D}$;
6. ISA is a binary relationship $\text{ISA} \subseteq \mathcal{C} \times \mathcal{C}$.
7. ID is a function, $\text{ID} : \mathcal{C} \mapsto \mathcal{A}$, that maps a class symbol in \mathcal{C} to its key attribute and $A \in \mathcal{A}$ is an attribute already defined in $\text{ATT}(C)$, i.e., $\text{ID}(C)$ may be defined only if $(A, D) \in \text{ATT}(C)$ for some $D \in \mathcal{D}$.

► **Definition 8** (\mathcal{DC}_p Semantics). *Let Σ be a \mathcal{DC}_p conceptual data model. An interpretation for the conceptual model Σ is a tuple $\mathcal{I} = (\Delta^{\mathcal{I}} \cup \Delta_{\mathcal{D}}^{\mathcal{I}}, \cdot^{\mathcal{I}})$, such that:*

- $\Delta^{\mathcal{I}}$ is a nonempty set of abstract objects disjoint from $\Delta_{\mathcal{D}}^{\mathcal{I}}$;
- $\Delta_{\mathcal{D}}^{\mathcal{I}} = \bigcup_{D_i \in \mathcal{D}} \Delta_{D_i}^{\mathcal{I}}$ is the set of basic domain values used in Σ ; and
- $\cdot^{\mathcal{I}}$ is a function that maps:
 - Every basic domain symbol $D \in \mathcal{D}$ into a set $D^{\mathcal{I}} = \Delta_{D_i}^{\mathcal{I}}$.
 - Every class $C \in \mathcal{C}$ to a set $C^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$.

Textual notation	Mapping to EER	Mapping to UML	Textual notation	Mapping to EER	Mapping to UML
C			ID		
R			CARD _R	$\frac{m..n}{m..n}$	$\frac{m..n}{m..n}$
U			CARD _A	$\frac{m..n}{m..n}$	
ATT			ISA		

■ **Figure 7** Sample mapping of the \mathcal{DC}_p textual elements from Definition 7 to graphical elements of one of the EER notational flavours and to UML Class Diagram elements.

- Every relationship $R \in \mathcal{R}$ to a set $R^{\mathcal{I}}$ of \mathcal{U} -labeled tuples over $\Delta^{\mathcal{I}}$ – i.e. let R be an binary relationship connecting the classes C_1, C_2 , $\text{REL}(R) = \{U_1 : C_1, U_2 : C_2\}$, then, $r \in R^{\mathcal{I}} \rightarrow (r = \{U_1 : o_1, U_2 : o_2\} \wedge \forall i \in \{1, 2\}. o_i \in C_i^{\mathcal{I}})$.
- Every attribute $A \in \mathcal{A}$ to a set $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta_{\mathcal{D}}^{\mathcal{I}}$, such that, for each $C \in \mathcal{C}$, if $\text{ATT}(C) = \{A_1 : D_1, \dots, A_h : D_h\}$, then, $o \in C^{\mathcal{I}} \rightarrow (\forall i \in \{1, \dots, h\}, \exists d_i. \langle o, d_i \rangle \in A_i^{\mathcal{I}} \wedge \forall d_i. \langle o, d_i \rangle \in A_i^{\mathcal{I}} \rightarrow d_i \in \Delta_{D_i}^{\mathcal{I}})$.

\mathcal{I} is said a legal database state or legal application software state if it satisfies all of the constraints expressed in the conceptual data model:

- For each $C_1, C_2 \in \mathcal{C}$: if $C_1 \text{ ISA}_C C_2$, then $C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$.
- For each $R \in \mathcal{R}$ with $\text{REL}(R) = \{U_1 : C_1, U_2 : C_2\}$: all instances of R are of the form $\{U_1 : o_1, U_2 : o_2\}$ where $o_i \in C_i^{\mathcal{I}}$, $U_i \in \mathcal{U}_i^{\mathcal{I}}$, and $1 \leq i \leq 2$.
- For each cardinality constraint $\text{CARD}_R(C, R, U)$, then:
 $o \in C^{\mathcal{I}} \rightarrow \text{CMIN}(C, R, U) \leq \#\{r \in R^{\mathcal{I}} \mid r[U] = o\} \leq \text{CMAX}(C, R, U)$.
- For each multiplicity constraint $\text{CARD}_A(C, A)$, then:
 $o \in C^{\mathcal{I}} \rightarrow \text{CMIN}(C, A) \leq \#\{(o, a) \in A^{\mathcal{I}}\} \leq \text{CMAX}(C, A)$.
- For each $C \in \mathcal{C}$, $A \in \mathcal{A}$ such that $\text{ID}(C) = A$, then A is an attribute and $\forall d \in \Delta_D^{\mathcal{I}}. \#\{o \in C^{\mathcal{I}} \mid \langle o, d \rangle \in A^{\mathcal{I}}\} \leq 1$.

► **Definition 9** (Mapping \mathcal{DC}_p into \mathcal{DLR}). Let $\Sigma = (\mathcal{L}, \text{REL}, \text{ATT}, \text{CARD}_R, \text{CARD}_A, \text{ISA}, \text{ID})$ be a \mathcal{DC}_p conceptual data model. The \mathcal{DLR} knowledge base, \mathcal{K} , mapping Σ is as follows.

- For each $A \in \mathcal{A}$, then, $A \sqsubseteq \text{From} : \top \sqcap \text{To} : \top \in \mathcal{K}$;
- If $C_1 \text{ ISA } C_2 \in \Sigma$, then, $C_1 \sqsubseteq C_2 \in \mathcal{K}$;
- If $\text{REL}(R) = \{U_1 : C_1, U_2 : C_2\} \in \Sigma$, then $R \sqsubseteq U_1 : C_1 \sqcap U_2 : C_2 \in \mathcal{K}$;
- If $\text{ATT}(C) = \{A_1 : D_1, \dots, A_h : D_h\} \in \Sigma$, then, $C \sqsubseteq \exists[\text{From}]A_1 \sqcap \dots \sqcap \exists[\text{From}]A_h \sqcap \forall[\text{From}](A_1 \rightarrow \text{To} : D_1) \sqcap \dots \sqcap \forall[\text{From}](A_h \rightarrow \text{To} : D_h) \in \mathcal{K}$;
- If $\text{CARD}_C(C, R, U) = (m, n) \in \Sigma$, then, $C \sqsubseteq \exists^{\geq m}[U]R \sqcap \exists^{\leq n}[U]R \in \mathcal{K}$;
- If $\text{CARD}_A(C, A) = (m, n) \in \Sigma$, then, $C \sqsubseteq \exists^{\geq m}[U]R \sqcap \exists^{\leq n}[U]R \in \mathcal{K}$;
- If $\text{ID}(C) = A \in \Sigma$, then, \mathcal{K} contains: $C \sqsubseteq \exists^1[\text{From}]A$; $\top \sqsubseteq \exists^1[\text{To}](A \sqcap [\text{From}] : C)$;

An example of this approach with a concrete CDM is illustrated in Example 2.

Standardizing Knowledge Engineering Practices with a Reference Architecture

Bradley P. Allen[†]   

University of Amsterdam, The Netherlands

Filip Ilievski^{†*}   

Vrije Universiteit, Amsterdam, The Netherlands

Abstract

Knowledge engineering is the process of creating and maintaining knowledge-producing systems. Throughout the history of computer science and AI, knowledge engineering workflows have been widely used given the importance of high-quality knowledge for reliable intelligent agents. Meanwhile, the scope of knowledge engineering, as apparent from its target tasks and use cases, has been shifting, together with its paradigms such as expert systems, semantic web, and language modeling. The intended use cases and supported user requirements between these paradigms have not been analyzed globally, as new paradigms often satisfy prior pain points while possibly introducing new ones. The recent abstraction of systemic patterns into a boxology provides an opening for aligning the requirements and use cases of knowledge engineering with the systems, components, and software that can satisfy them best, however, this direction has not been explored to date. This paper proposes a vision of harmonizing the best practices in the field of knowledge engineering by leveraging the software engineering methodology of creating reference architectures. We describe how a reference architecture

can be iteratively designed and implemented to associate user needs with recurring systemic patterns, building on top of existing knowledge engineering workflows and boxologies. We provide a six-step roadmap that can enable the development of such an architecture, consisting of scope definition, selection of information sources, architectural analysis, synthesis of an architecture based on the information source analysis, evaluation through instantiation, and, ultimately, instantiation into a concrete software architecture. We provide an initial design and outcome of the definition of architectural scope, selection of information sources, and analysis. As the remaining steps of design, evaluation, and instantiation of the architecture are largely use-case specific, we provide a detailed description of their procedures and point to relevant examples. We expect that following through on this vision will lead to well-grounded reference architectures for knowledge engineering, will advance the ongoing initiatives of organizing the neurosymbolic knowledge engineering space, and will build new links to the software architectures and data science communities.

2012 ACM Subject Classification Computing methodologies → Knowledge representation and reasoning; Software and its engineering → Software architectures

Keywords and phrases knowledge engineering, knowledge graphs, quality attributes, software architectures, sociotechnical systems

Digital Object Identifier 10.4230/TGDK.2.1.5

Category Position

Related Version *Full Version*: <https://arxiv.org/abs/2404.03624>

Received 2023-09-19 **Accepted** 2024-03-05 **Published** 2024-05-03

Editors Aidan Hogan, Ian Horrocks, Andreas Hotho, and Lalana Kagal

Special Issue Trends in Graph Data and Knowledge – Part 2

[†] Equal contribution

* Corresponding author



1 Introduction

Knowledge engineering (KE) is the collection of activities for eliciting, capturing, conceptualizing, and formalizing knowledge to be used in information systems [72]. KE includes two broader tasks of creating and maintaining knowledge [3]. Throughout the history of computer science and AI, KE workflows have been a critical component when building reliable intelligent agents across domains and tasks [53]. Indeed, it has been intuitive that developing trustworthy models for applications, from common sense to traffic, crime, and weather, requires well-understood knowledge processes [51]. Similarly, task solutions within these domains, including question answering, summarization, and forecasting, are expected to incorporate standardized KE procedures to be meaningfully applicable and compatible with humans [89].

The importance of knowledge production processes has yielded many notable architectures over the past decades, which aim to synthesize dominant patterns and best practices. As apparent from these architectures, the dominant paradigm of KE has been shifting through the history of AI, from its early days through the eras of expert systems and semantic web. Expert system workflows, like CommonKADS [77], enable the extraction of expert knowledge into knowledge bases based on lifecycle analysis and corresponding models. Many Semantic Web applications can be aligned to a general layered template, most famously the Semantic Web Layer Cake and its contemporary variants [31, 39, 9, 50]. Knowledge graph (KG) workflows [84] and comprehensive toolkits [43] aim to bridge the gap between knowledge bases and their applications, showing a larger emphasis on extensional and possibly less precise modeling of knowledge [79]. Knowledge graph engineering (KGE) emerged as a variant of KE geared towards capturing, representing, and utilizing complex information about entities, their relationships, and their underlying semantics [79, 33]. Researchers and domain experts have devised KGE workflows that are tailored to the needs of a variety of domains like biomedicine [54], library and information sciences [87], web democracy [90], commonsense knowledge [44], and publications [68]. Analogously, enterprise infrastructures, such as the Amazon Product Knowledge Graph [98], have been devised for commercial settings, without clear reference to a standardized workflow. The recent trends in KE, such as the consideration of large language models (LLMs) as knowledge artifacts [64] and the prominence of neurosymbolic systems [92, 72], bring a new perspective to KE. The role of LLMs in KE workflows is studied actively to understand the potential of LLMs to enhance, replace, or add KE components [34]. Meanwhile, recent work based on abstracting semantic web and machine learning systems (SWeMLS) [21] indicated the prominence of KE in neurosymbolic systems, with around a quarter of all SWeMLS patterns corresponding to a KE process.

The present landscape of KE methodologies and tools lacks a comprehensive framework of user needs and available paradigms, as each subsequent KE era does not necessarily include the benefits brought by its predecessors [3]. KE systems require a principled way of considering different user requirements, paradigms, and use cases, thus combining human, social, and technical factors [40]. To address this need, recent work has proposed the formalism of a *boxology*: a hierarchical taxonomy of systemic design patterns expressed in a graphical notation (cf. Figure 2 and the upper right hand corner of Figure 4) [92, 72]. Boxologies provide an opening for aligning the requirements and use cases of knowledge engineering with the systems, components, and software that can satisfy them best, however, this direction has not been explored to date. We see an urgency to understand the scope and purpose of KE in the latest evolving AI landscape, aiming to devise a general framework characterized by requirement-driven best practices. Such a framework should ideally support the perspectives of existing and emerging KE paradigms, enable a flexible definition and support for stakeholder requirements and priorities, and build on top of prior work on KE workflows and systemic patterns. Given the dynamic nature of KE, it should

also provide mechanisms to adapt to evolving requirements over time and across applications. It should provide a prescriptive framework that standardizes best practices while allowing them to be customized to specific circumstances.

This paper proposes a vision of harmonizing the field of knowledge engineering by leveraging the software engineering methodology of devising a reference architecture (RA), inspired by successful RAs designed and applied in domains such as the automotive sector, e-government, and service-oriented solutions [29]. RAs serve as a framework that standardizes a community of practice through a software engineering artifact, based on a survey of relevant stakeholders, their requirements, existing community-based workflows, and suitable evaluation. RAs are developed in a human-centric and iterative manner, which makes them particularly suitable for dynamic and frequently changing disciplines such as KE. They provide a common framework, while simultaneously enabling users to design specific RAs for their narrow use cases. The proposal to develop a methodology for devising RAs for KE is in line with the suggestions in [40]: we hypothesize that the development of RAs will make KE and related knowledge technologies accessible for outsiders and newcomers from disciplines such as software engineering and data science that have compatible goals.

We consider how RAs can be iteratively designed and implemented to associate user needs with recurring systemic patterns, building on top of existing KE workflows and boxologies. Section 2 provides an extended problem statement that motivates the need to consolidate KE practices by building on top of existing boxology patterns. Section 3 provides relevant background on existing RAs and common methodologies for their principled development, and reviews state-of-the-art architectures for KE. Section 4 details a six-step methodology to design and implement a human-centric reference architecture that standardizes KE practices, consisting of scope definition, selection of information sources, architectural analysis, synthesis of an architecture based on the information source analysis, evaluation through instantiation, and, ultimately, instantiation into a concrete software architecture. We describe an initial design of the architectural scope, information sources, and analysis, and prescribe the processes for the design, evaluation, and instantiation of the architecture, as the latter steps are highly use-case dependent. The paper is concluded in Section 5. We expect that following through on this vision will lead to well-grounded reference architectures for knowledge engineering, and will facilitate further links to the software architectures and data science communities.

2 Why a Reference Architecture Framework for KE?

The pursuit of a general reference architecture framework for KE is motivated in this section by three key factors. First, the KE paradigms have been shifting over time, each following one addressing pain points of the existing paradigms, but often failing to address other requirements. Second, KE users vary greatly, and while the user needs have been often discussed in the context of a specific application, a broad view of connecting users, their tasks, and their corresponding requirements is lacking. Third, the emerging boxology of neurosymbolic systems, with its recent link to knowledge engineering, provides a unique opportunity to exploit emerging patterns as components of a more comprehensive architecture.

2.1 Historiographic perspective: Consolidation of the KE paradigms

Prior research on KE is rich, spanning from the 1950s, through the expert systems era of the 1980s, the Semantic Web era, and the recent view of language modeling as a knowledge production process [70, 57, 24, 25, 77, 11, 64, 40, 43, 10, 36, 86, 2]. These different periods have approached KE following the contemporary technological, scientific, and societal focus. We provide a brief historiographic view on KE here, for an extended discussion we refer the reader to [3].

5:4 Reference Architectures for Knowledge Engineering

In the 1960s, researchers like Newell and Simon [57] were hopeful about the ability of goal-directed search with heuristics to perform practical general-purpose problem-solving. However, by the 1970s, it became evident that these systems were not easy to scale to complex applications. During the mid-1970s, Feigenbaum [24], influenced by Newell and Simon's work, maintained that focusing on specific domains was crucial for successful knowledge engineering. Knowledge engineers worked on the elicitation of domain-specific knowledge with high quality and expressivity, and domain-independence and scalability were often not prioritized. This period saw a surge in creating expert systems for decision support in businesses, but by the early 1990s, it was clear that these systems had limitations, being brittle and hard to maintain. Efforts to address these limitations included the development of structured methodologies for knowledge engineering during the late 1990s [77]. Feigenbaum [25] persisted in exploring the idea of domain-specific applications but suggested that future systems should be scalable, globally distributed, and interoperable. These ideas, ahead of their time, foreshadowed some aspects of what later became the World Wide Web.

In the era of the Semantic Web, Tim Berners-Lee [11] advocated the use of specific open standards (e.g., RDF and SPARQL) to encode knowledge in Web content, to improve access and discoverability of Web content, and to enable automated reasoning. However, adoption of Semantic Web technology was slow, ultimately leading researchers to seek ways to align these standards and principles more closely with general software industry norms and make them more developer-friendly. Recent efforts, particularly in commercial knowledge graphs developed by companies like Google and Amazon [98], have shown a shift towards custom architectures, often based on property graphs. This shift, while innovative, often sidesteps the interoperability and federation ideals of early visionaries like Feigenbaum and Berners-Lee. As a result, there's a growing need to refine what KE offers developers, focusing on comprehensive, scalable, customizable, and modular infrastructures that integrate with common data formats. KE should be domain-independent, supporting a wide range of use cases with user-friendly interfaces.

The rise of connectionist methods and graphical processing hardware in the 2010s has introduced new possibilities for knowledge production using large language models (LLMs). LLMs have been shown to be a means to provide robustness to missing schema and better generalization across domains and knowledge types. Two main perspectives have emerged regarding the relationship between LLMs and knowledge bases [2]. The first sees LLMs as standalone, queryable knowledge bases that can learn from unstructured text with minimal human intervention [64]. This method challenges traditional, labor-intensive KE processes, but raises concerns about accuracy, ethical use, interoperability, and curatability. The second, more cautious perspective views language models as components in a KE workflow, combining new and traditional methods [43]. This approach emphasizes accessibility, manual editing of extracted knowledge, and explanation of reasoning methods, addressing the limitations of earlier technologies. Both perspectives highlight the importance of sustainability and affordability in KE processes.

2.2 Social perspective: Systematic procedures for incorporating stakeholder tasks and needs

KE tasks can be roughly split into two main categories in terms of their goal: creating and maintaining knowledge artifacts. Here, the knowledge artifacts are typically knowledge graphs, ontologies, and taxonomies [72]. Representative KE tasks are shown in Table 1. These include tasks of creating an ontology or refining that ontology, for example, to include a newly discovered concept or relationship [58, 26, 83]. KE tasks include ingesting and transforming data from multiple data sources into a single artifact, or performing data integration between different schemas [45, 19]. Resulting KGs can be further refined to address issues such as inconsistencies of modeling, contradictions of factual information, outdated information, or missing/incomplete

■ **Table 1** Representative user tasks and scenarios for knowledge engineering.

Task	Scenario
Ontology creation	A new domain is identified, for which an ontology needs to be created.
Ontology refinement	A new concept or relationship is identified in the domain, and the ontology needs to be modified to support it without disruptions.
Data ingest and transformation	Multiple data sources provide overlapping or complementary information. The system needs to transform and normalize this data to ensure consistency in the knowledge graph.
Data source integration	A new data source, in a previously unsupported schema, needs to be incorporated into the knowledge graph while ensuring data quality.
Anomaly detection	The system flags a potential inconsistency or contradiction in the knowledge graph, which needs to be resolved.
Knowledge graph completion	The system flags a missing or incomplete statement, which needs to be automatically inserted.
Human oversight of knowledge graph quality	A subject matter expert (SME) identifies a piece of outdated or incorrect information in the knowledge graph, which needs to be flagged to initiate a correction.
Human feedback	As SMEs interact with the system, they might have insights or suggestions based on their domain expertise, which needs to be supported and incorporated into the refinement process.

statements, to improve their quality [63]. Such issues may be raised by automated systems [78] as well as human subject matter experts (SMEs) [65]. Finally, humans interacting with the knowledge artifact may have further suggestions or feedback for refinement [43].

Commonly, KE procedures identify local requirements for an artifact, with an implicit assumption of the user profile. Meanwhile, user studies are increasingly present in KE research [1, 42], a trend that can be enriched by considering the natural plurality of users. While early KE might have been carried out by computer science practitioners, today it often includes domain experts interacting with knowledge directly, knowledge engineers building ontologies, knowledge editors fixing outdated information, data scientists developing knowledge completion systems, and business/organizational stakeholders that stress-test the available knowledge to understand its value [42]. Considering the example tasks in Table 1, we note that the tasks of ontology creation and data integration require expertise from knowledge engineers and subject matter experts. Refining of knowledge artifacts can be performed by knowledge engineers or data scientists, whereas providing human feedback and oversight requires subject matter experts. Many of these tasks may also benefit from the inputs from business and organizational stakeholders. While here we refer to *stakeholders* as humans that create and maintain knowledge engineering processes, there is an additional set of tasks and users that *use* the artifact resulting from the knowledge engineering process, such as data scientists developing AI prototypes that reason over knowledge and software developers that build knowledge-infused chatbots.

2.3 Component perspective: Preliminary source of architectural patterns

Driven by societal and technical needs for explainability, robustness, and collaboration, neurosymbolic AI has been growing in popularity recently, emerging as one of the key trends of AI research [46, 72]. Each neurosymbolic system combines neural, machine-learning components with

symbolic manipulation. Given the breadth of this definition, there have been attempts to organize neurosymbolic systems by abstracting their architectures using recurring patterns. Following the *boxology* approach [92, 72], data structures can be symbols or data, whereas algorithmic modules are either inductive (machine learning) or deductive (based on knowledge representation and reasoning formalisms). Then, each boxology pattern is a combination of alternating data structure and algorithmic module boxes. The initial boxology work identified 15 such patterns. 11 of these, together with 33 new patterns, were found in the systems systematically surveyed in [14]. The 44 patterns have been classified into a pattern typology based on their complexity, e.g., simple patterns have a single processing unit. Sample patterns from this boxology are illustrated in Figure 2, which we describe in more detail in Section 4.

The question emerges: what is the role of KE in NeSy AI systems? How often do NeSy architectures represent KE processes? An insight into these questions is provided by [72], who coined the term *neurosymbolic knowledge engineering* and analyzed the NeSy approaches that combine machine learning and semantic web components. While we see such component analysis as a step in the right direction of organizing neurosymbolic KE, we identify three challenges for state-of-the-art KE that are apparent from the boxology framework. *Challenge 1:* The patterns should be associated with user requirements, tasks, and application needs to enable their efficient and precise application. The present boxology patterns do not include this information, nor have the mechanism built in to include it in the future. *Challenge 2:* Mechanisms for aligning with ongoing trends and shifting requirements are lacking. These are needed as the set of boxology patterns and their specification (e.g., type constraints) are still largely in flux, as apparent from the large number of newly discovered patterns in [14], and given the lack of specification of how the boxology primitives align with popular NeSy processes (e.g., fine-tuning) and artifacts (e.g., knowledge graphs). *Challenge 3:* There is a lack of a standard for communicating the KE boxology patterns to non-knowledge engineers, including software engineers, data scientists, domain experts, and business stakeholders. While the abstraction of the boxology patterns makes a step towards facilitating broader comprehension, the patterns are still meant for experts.

3 Related Work

The previous section discussed three considerations that motivate the need for a reference architecture framework for the standardization of KE practices. This section summarizes definitions, practices, and methodologies associated with work on RAs, and describes how research in the areas of data, knowledge, and ontology engineering can contribute to the establishment of reference architectures for KE, towards the end of consolidating the three motivating perspectives.

3.1 Reference architectures

Definition and uses A reference architecture is a framework that aligns stakeholders' requirements with design patterns through a final architecture and a corresponding software system. As such, an RA serves as a generic architecture for a class of information systems within a software engineering community of practice [5]. RAs have several shared characteristics: they provide the highest level of abstraction, they heavily emphasize architectural qualities, their stakeholders are considered but absent from the architecture, they promote adherence to common standards, and are effective for system development and communication [7]. Notably, while architectures capture software structures, not every structure is architectural: architecture is an abstraction that should emphasize the attributes that are important to stakeholders [8]. For a comprehensive review of software RAs, we refer the reader to [29].

RAs are driven by two emerging trends [18]. First, an increasing complexity, scope, and size of the system of interest, its context, and the organizations creating the system. Second, increasing dynamics and integration, i.e., shorter time to market, more interoperability, rapid changes, and adaptations in the field. In [8], the authors identify thirteen uses for developing a central reference architecture. A key aspect of reference architectures, along with related types of architectural artifacts, is that they are key to the creation of a technology strategy that drives consensus across multiple groups of stakeholders with an enterprise engaged in software application development for business purposes. Other benefits include that RAs enable the system's quality attributes, enable early prediction of system qualities, encode fundamental design decisions, support the training of new team members, reduce system complexity, and facilitate reuse. Reference architectures provide a common lexicon and taxonomy, a common architectural vision, and modularization [18]. Notably, good architecture is necessary but not sufficient to ensure quality.

Many RAs have been proposed in the past decades, some of which have gained wide adoption in their domains. Well-known examples are AUTOSAR for automotive sector [81], CORBA for object integration through brokers [12], S3 for service-oriented solutions [6], EIRA for e-Government systems,¹ and NIST's Big Data Interoperability Framework [27]. We describe AUTOSAR in greater detail to provide an example of a typical RA. First introduced in 2003, AUTOSAR was developed as a cooperative effort between major automotive manufacturers, suppliers, and tool developers. The primary goal of AUTOSAR is to enable the development of highly modular, scalable, and reusable software components for automotive applications. By providing a common software infrastructure and standardized interfaces, AUTOSAR aims to reduce development costs, improve software quality, and facilitate the integration of software components from multiple suppliers.

Ironically, while the field of Semantic Web puts a lot of emphasis on developing artifacts like ontologies and knowledge graphs that enable common understanding between humans and machines, it has not caught up on the idea of developing architectures, such as AUTOSAR, that will provide a common framework in which different concerns can be expressed, negotiated, and resolved among stakeholders for large, complex knowledge systems [8].

Methodologies for creating RAs. A method to design a software architecture has been proposed by [56], consisting of five steps: establishing its scope, selecting and investigating information sources, performing an architectural analysis to identify architecturally significant requirements, carrying out synthesis of the reference architecture, and evaluating the architecture through surveys as well as its instantiation and use. Typical RAs for big data usually follow a three-step lifecycle consisting of data ingestion, transformation, and serving [7]. Their major architectural components can be roughly grouped into 1) big data management and storage, 2) data processing and application interfaces, and 3) big data infrastructure. Two types of requirements are commonly used to describe stakeholder needs for such software architectures: functional requirements (FRs) and quality attributes (QAs) [8]. *Functional requirements* typically describe what the system components are responsible for, i.e., they state what the system must do and how it must behave or react to runtime stimuli. They are satisfied by assigning an appropriate sequence of responsibilities throughout the architectural design. *Quality attribute* is “a measure or testable property of a system that is used to indicate how well the system satisfies the needs of its stakeholders.” Quality attributes must be characterized using one or more scenarios, and they must be unambiguous and testable.

¹ <https://joinup.ec.europa.eu/collection/european-interoperability-reference-architecture-eira/about>

An example of the application of RAs to knowledge engineering is the work of Ocaño et al. on an RA for integrating artificial intelligence and knowledge bases to support journalists and newsrooms [61]. They apply a methodology similar to that of [56], taking domain-specific requirements for the effective support of journalistic activities, defining a reference architecture, and then implementing a prototype instantiation of that architecture. This architecture provides a crucial example of what a realization of an RA for KE would look like for a particular domain. This paper provides a streamlined procedure for instantiating other procedures based on a general RA framework. In the case of general RAs for KE, prior work has devised a set of QAs and FRs [3] based on a historiographic analysis. The present paper considers how this effort can be advanced to result in a general-purpose RA framework for KE.

3.2 Methodologies and workflows for knowledge engineering

Reference architectures can serve as a framework that shapes and optimizes knowledge engineering workflows, ensuring they are efficient, scalable, and compliant with best practices and standards; conversely, knowledge engineering methodologies and workflows can drive the definition of RAs by providing structured approaches to requirements specification and providing specific choices of technologies that constrain the design of a reference architecture.

Knowledge engineering. From the earliest days of the expert systems era there was a recognition that KE needed a principled methodology [38], but the first complete realization of such a methodology came in the 1990s with the development of KADS [94] and subsequently CommonKADS [77]. CommonKADS is a methodology for the extraction of expert knowledge into knowledge bases based on lifecycle and corresponding models. CommonKADS has been applied to a variety of domains, from e-governance [96] to multi-agent scenarios [41]. The models formalized by CommonKADS are complemented by MIKE's [4] formalization of the execution of the model, and the Protege [30] software for collaborative knowledge production and maintenance. The primary focus of this work is on aspects of task selection, knowledge modeling, and knowledge elicitation, and relatively little attention was paid to architectural aspects and deployment in modern Web-based applications and services, except for the linked data community's emphasis on the use of W3C linked data stack and standards [39]. More recently, the growth of Semantic Web applications has resulted in research into semantic patterns [28] and boxologies that organize systems using abstract components [92]. While these boxologies originally aimed to capture purely automated processes, there have been attempts to include human agents, either as process initiators [91] or following the human-in-the-loop paradigm [95]. With the emergence of knowledge graphs, recent work has devised corresponding workflows for particular domains like the Library and Information Studies (LIS) [87] community, e-commerce applications like the Amazon Product KG [98], and generic workflows for the biomedical domain [55]. Finally, there have been attempts to identify common patterns in knowledge graph workflows [84] and design toolkits [43] that implement these patterns as reusable pipelines of commands.

Ontology engineering. A specific area of focus within knowledge engineering is ontology engineering (OE) [32]. The Semantic Web era is characterized by a strong focus on the manual development of ontologies [58] and their publishing on the Web using linked data principles, with a strong focus on interoperability, reuse, and integration [66]. Methodologies for ontology engineering developed over the past thirty years include METHONTOLOGY [26], Kendall and McGuinness's Ontology Development 101 [48], and NeOn [83]. As with the KE methodologies described in the previous section, OE methodologies are concerned with the organizational structures and workflows associated with ontology design, knowledge representation (e.g. the modeling of spatio-temporal

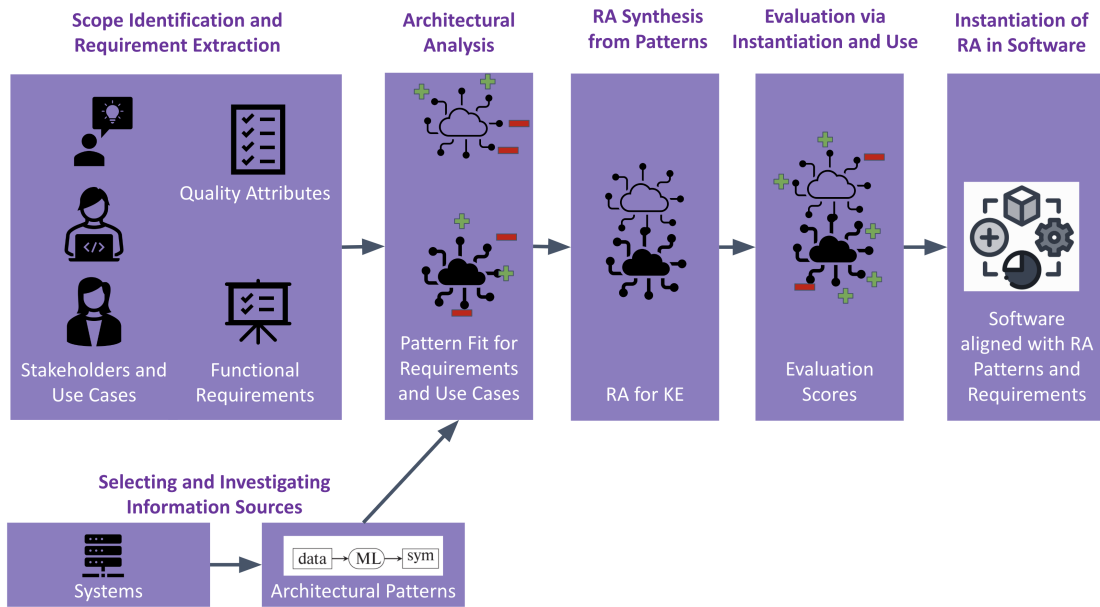
modeling [35, 23]), and ontology matching [62]. There has been limited work in understanding the relationship between data governance [49] and ontology engineering; while both disciplines overlap in their concerns for structuring and managing data, the integration of data governance principles into ontology engineering workflows remains a less explored area. OE and KE workflows are abstracted through the neurosymbolic boxology patterns [72], which enables them to be represented in an overarching architecture that is composed of those patterns.

Data engineering. Data engineering (DE) itself has provided a wide range of best practices and workflows in common use across the industry. Standard architectural models of data processing systems include data warehouses [16], data lakes [71], and distributed data processing platforms such as Apache Spark [74]. Recent work on *DataOps* [22] as an adaptation of DevOps principles and best practices to the design and operation of data processing workflows has established many concepts towards the end of ensuring that data ingestion and integration are smooth, continuous, and error-free. These principles include the monitoring of the quality of data to prevent poor quality or inconsistent data from compromising data integrity of the knowledge graph; data versioning, supporting the ability to revert to previous states of the data or understand changes over time; and designing workflows such that the system can scale accordingly without a compromise in performance [7, 82]. All of these techniques can inform the design of architectures for knowledge engineering. The Andreessen Horowitz reference architecture [13] for emerging data infrastructure and platforms is a snapshot of the current industry stack and trends that subsume most current uses of data within an enterprise. This architecture includes several high-level elements, such as sources, ingestion and transport, storage, query and processing, transformation, and analysis and output. It is noteworthy that, while this architecture has been adapted for artificial intelligence and machine learning workflows, it does not refer at all to knowledge graphs or Semantic Web concepts or products, especially given the care it takes to address specific use cases related to machine learning.

4 A six-step roadmap to an RA for KE

By using a requirements-driven approach [8, 5, 18], RA methodologies, informed by recent work on DE, KE, and OE methodologies and workflows, can support the consolidation of different perspectives and paradigms under a single umbrella (*challenge 1*). RAs provide a suitable approach for technological alignment by first identifying, consolidating, and prioritizing user needs, followed by formalizing these needs into functional requirements and quality attributes, and, finally, following an iterative development and evaluation of architectures that satisfy these requirements best. Addressing *challenge 2*, using a requirements-driven iterative design, RAs are developed to suit current technological trends and to be dynamically adapted in the future when the underlying requirements shift significantly. In other words, RAs are designed to be representative of the current technological trends and are flexible to be enhanced over time to suit further developments that are likely to occur in a dynamic field such as KE. As mainstream software engineering artifacts, RAs can facilitate smoother adoption of KE by software engineers and computer/data scientists (*challenge 3*). An RA is a mechanism for meeting practitioners in such fields halfway and enabling a bridge for seamless integration and collaboration between these fields and KE.

We propose that RAs for KE should be designed by applying the mainstream software engineering techniques described in the previous section. We adopt the methodology proposed by Nakagawa et al. [56], consisting of five steps: scope identification (including extraction of requirements), selecting and investigating information sources, architectural analysis, synthesizing



■ **Figure 1** Pipeline for devising an RA for KE. First, we identify the scope by defining stakeholders and use cases, ultimately resulting in a set of quality attributes and functional requirements [3]. Second, we select and investigate information sources, according to the SWeMLS corpus of neurosymbolic systems and patterns for KE [21, 72]. Third, we connect these components through architectural analysis, yielding information about the fit of various patterns for requirements and use cases. Based on these insights, the fourth step synthesizes an RA from these patterns. Fifth, the RA is evaluated through instantiation and use using a standard software architecture methodology. Finally, the RA is instantiated into software.

an RA, and evaluating the RA through instantiation and use. We include an additional step of instantiating the RA in software, resulting in a six-step procedure. We see the last three steps as iterative steps, which can be modified given the shifting stakeholder requirements, the modular design of the architecture, and the dynamic nature of the underlying technology for the software implementation. The methodology for devising an RA for KE is summarized in Figure 1.

4.1 Scope identification and extraction of requirements

We take inspiration from the software engineering practice of using reference architectures as consolidation mechanisms. On the one hand, the RA framework needs to cover the **use cases** that fall under the task of KE. According to our definition and following [72], KE is a knowledge process that includes knowledge creation (e.g., ontology creation, data ingest) and refinement (e.g., ontology refinement, knowledge graph completion, anomaly detection). The scope of the RA framework should enable machine, human, and joint machine-human knowledge processes [89]. The set of tasks that fall within the scope of KE are listed in Table 1, together with a typical scenario and a question that an RA should be designed to solve. For example, the knowledge graph refinement task can be illustrated with a system flagging a potential inconsistency or contradiction. A question for an RA is how it can facilitate the resolution of such quality challenges.

On the other hand, the RA framework must identify and support the **requirements** of the relevant stakeholders. In software architecture development [18, 85, 8], requirements serve as a common denominator to align the needs of the stakeholders and the technical patterns. While stakeholders may include both knowledge engineers and beneficiaries of KE (e.g., data

scientists building applications), we focus on the requirements of knowledge engineers, i.e., users that perform the aforementioned knowledge graph creation and refinement tasks. As is common in software engineering [8], the requirements can be translated into two categories: functional requirements and quality attributes. In recent work [3], we devised a set of 23 quality attributes and 8 functional requirements for KE, based on a historiographic analysis of the field of KE (similar to Subsection 2.1). We show an excerpt of five quality attributes and five functional requirements in Table 2. These requirements are manually selected to show diverse representative FRs and QAs. Their role is to illustrate to the reader what FRs and QAs for KE (may) look like. An example of QA is modularity, namely, a requirement that the components of the KE workflow enable selective composition for supporting particular use cases. An example of an FR is the import of common data formats, including mainstream semantic web and software sources, as well as serializations. While we consider [3] to provide an initial set of FRs and QAs, we note that the review of papers in this prior work is not based on a systematic selection. The work on analyzing Semantic Web and Machine Learning Systems (SWeMLS) identifies three other requirements: maturity, transparency, and auditability, based on a systematically collected set of papers [72]. Critical future work is to explore how to automatically derive FRs and QAs from such a systematically collected set of papers, based on formally defined requirements. Moreover, given a particular, more narrow scope, the users are expected to define a subset of high-priority requirements that will guide the construction of their RA.

4.2 Selection and investigation of information sources

A systematic analysis of the NeSy landscape, aiming to characterize SWeMLS published between 2010 and 2020, resulted in a corpus of 476 system papers [14]. In this work, each of the papers was annotated with bibliographic information (authors, institutions, publication year, and venue), domain of application, task solved, input/output system architecture, characteristics of the machine learning and the semantic web modules, and levels of maturity, transparency, and provenance. The system components are aligned to the boxology for neurosymbolic systems [92]. In total, 44 patterns were discovered, classified into a typology of six types according to their shapes. Some example boxology patterns from the SWeMLS corpus are shown in Figure 2. The F2 pattern (short for *fusion-2*) is described in [93] as a simple fusion design pattern that takes both symbolic (s) and unstructured data (d) as inputs and produces symbolic data (s) as output using a model M. The F2 pattern corresponds to two specific systems, one being a geological text document classifier [69], and the other an application that classifies heterogeneous web content to create symbolic data extending an enterprise knowledge graph [80].

The data from the study by [21] is made available as a knowledge graph. The ontology of this knowledge graph is centered around the class `System`, which belongs to one `Pattern` and has N `System Component` values. Using SPARQL queries against the SWeMLS knowledge graph, we identified a subset of 139 papers as KE-related, consisting of papers whose systems perform `Graph creation` or `Graph extension` tasks, and produce `Symbol` as the final output. In doing so, we followed the procedure described in [72]. We use this set of 139 KE papers in the rest of our methodology, given the systematic approach to collecting them, their rich annotation, and their alignment with the NeSy boxology components. This procedure illustrates how the selection of information sources can be achieved - in practice, RA developers may decide to focus on a different set of sources, e.g., covering a larger set of papers or a particular subarea of KE for better representativeness to their envisioned use cases.

■ **Table 2** Example QAs and FRs for KE from [3], extended with evaluation criteria. We refer to each requirement with “should” signifying a uniform level of importance. The priority scale of the requirements can be further distinguished according to the specific use case requirements.

requirement	description	evaluation criteria
interoperability [25]	the knowledge produced by the KE process should be easy to share across sites and applications	compatibility with different data formats and standards; ease of integration with other systems; number of supported interfaces/APIs
curatability [10]	the KE process should support human curation of automatically extracted and/or inferred knowledge	effectiveness of human curation interfaces; balance between automation and human oversight; quality control measures for curated knowledge
scalability [25]	the KE process should scale economically with the amount of knowledge produced (measured in terms of rules, triples, nodes, edges, etc.)	performance under increasing amounts of knowledge (e.g., response times, throughput); cost-effectiveness at different scales; system behavior under concurrent user loads
modularity [43]	the components of the knowledge engineering process should be selectively composable to suit a specific use case	independence and interchangeability of system components; ability to integrate or detach modules based on need; impact of module changes on overall system performance
customizability [43]	the components of the KE process should be modifiable to support specific use cases	ease and extent of system modifications; number of customizable components; user feedback on customization features
supports semantic web standards [11]	the KE process should support the use of W3C semantic web standards	use of standard knowledge representation (e.g., RDF, property graphs), serializations (e.g. Turtle, JSON-LD) and query languages (e.g., SPARQL, Cypher), evaluated by ontology quality metrics, pitfall scanning
imports common data formats [43]	the KE process should support the import of data and/or knowledge from data sources	use of standard serializations (e.g., CSV, JSON, Parquet), evaluated by ontology quality metrics, parsing error rate
exports common data formats [43]	the produced knowledge should be exportable to software industry-standard data delivery mechanisms	use of software industry-standard data storage mechanisms (e.g., relational databases, RDF data dumps, search engine indexes) and integration standards (e.g., serialized data dumps, publish/subscribe messaging, REST APIs), evaluated by time to deploy, storage and compute costs
provides user-friendly interfaces [43]	the knowledge produced by the KE process should be accessible and applicable by end users	industry-standard user experience (e.g., command line interfaces, visual editors and browsers, reporting and analytics dashboards) measured by time to complete tasks, user satisfaction surveys
supports heterogeneous query [36]	the knowledge produced by the KE process should be searchable using multiple query languages	use of industry-standard query languages (e.g., SQL, Cypher, SPARQL) and query execution strategies (e.g., federated query, centralized query, find-and-follow), with developer experience measured by time to complete tasks, user satisfaction surveys

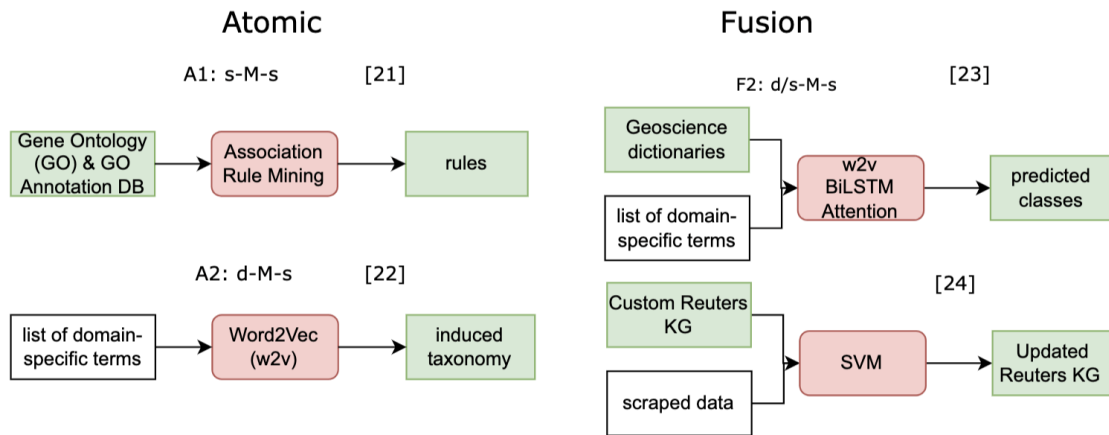


Figure 2 Simple neurosymbolic system design patterns from the SWeMLS KG, as shown in [93]. The F2 design pattern, appearing on the right of the figure, is a simple fusion that takes both symbolic (s) and unstructured data (d) as inputs and produces symbolic data (s) as output using a model M.

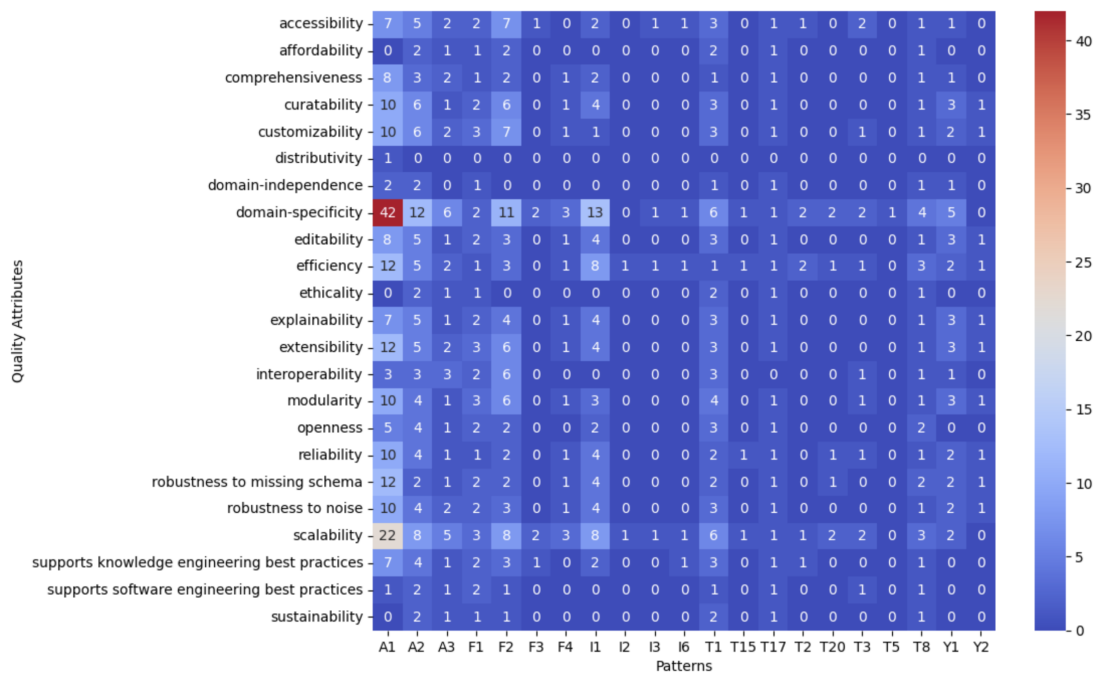


Figure 3 Preliminary analysis of the relationships between quality attributes for KE identified in [3] and the KE design patterns from [72] that are associated with knowledge graph creation and extension. The number in each cell is the count of occurrences of the quality attributes assigned to papers by the zero-shot text classifier that describes systems with the given pattern.

4.3 Architectural analysis

The next step is to perform a preliminary analysis of the extent to which quality attributes for KE are supported within specific SWeMLS patterns. We illustrate this analysis over the SWeMLS KG, where papers describe a system, and each system is associated with a specific pattern. To

establish a connection between quality attributes and patterns, we utilize the SerpApi Google Scholar API to obtain snippets from the abstracts of each of the 139 papers described in the previous section.² We then construct a zero-shot text classifier using prompt programming of ChatGPT [75] that, given an article’s snippet and title, assigns one or more quality attributes to each paper. Here, we used the 23 QAs identified in the first step. We then aggregate the quality attributes for each paper’s system’s pattern across all of the papers and patterns. This allows us to derive a **matrix relating quality attributes to patterns**, as shown in Figure 3. From this initial analysis, we find that the A1, A2, F2, and T1 patterns cover the most quality attributes. Namely, A1 covers 20 of the 23 QAs, except for affordability, ethicality, and sustainability; A2 and T1 only lack the QA of distributivity; and F2 covers 20 QAs lacking only distributivity, domain independence, and ethicality. We find these insights to be largely intuitive, as many systems belong to patterns such as A1 and F2. Among the quality attributes, we note that most patterns capture domain-specificity and scalability, whereas ethicality and distributivity are rare. This indicates the tendency of neurosymbolic KE systems to focus on scalability and domain-specificity, whereas aspects such as sustainability, ethicality, and distributivity are gradually gaining momentum but are not yet a primary consideration for most systems.

We emphasize that the corpus used for our analysis is not comprehensive and that the specific analytical methodology followed in this paper may exhibit classification bias. Thus, the significance of this analysis is mainly to show an illustration of how architectural patterns and quality attributes can be linked together. This provides us with a means to determine, given the quality attributes and functional requirements from the scope identification and requirements extraction steps, which pattern(s) are candidates for RA synthesis. We leave it to future work to further tune this procedure, generalize it to a larger dataset, and devise a more robust classification engine. Finally, we note that an analogous procedure can be followed for aligning functional requirements with boxology patterns.

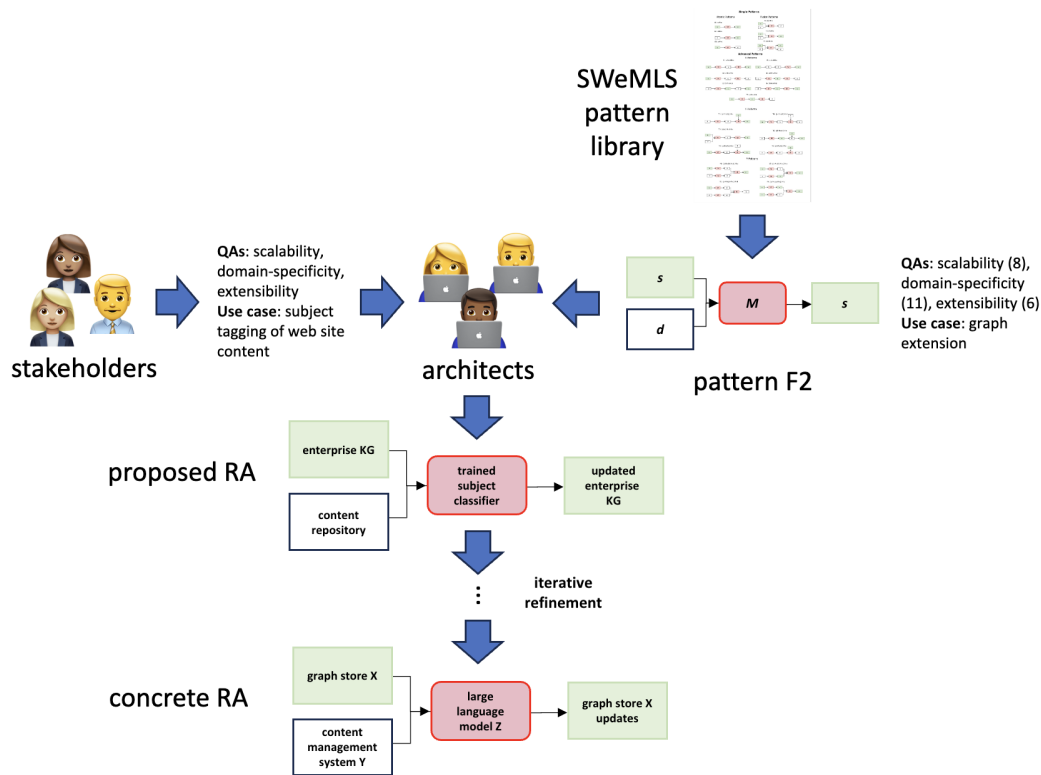
4.4 RA synthesis from patterns

4.4.1 Procedure

Given the architectural analysis of the patterns from the boxology and from other prominent workflows, the construction of the RA follows as a natural synthesis step. Namely, the RA consolidates the discovered pattern(s) with consideration for their adequacy for addressing the use cases and the derived requirements. The benefit of this synthesis is that it prescribes a global view of how a given pattern addressed the stakeholder needs, how the different patterns fit together if a complex pattern is being composed of simpler patterns, and how the architectural pattern(s) can be technically realized; all of that, while aligning with state-of-the-art workflows as reported in the literature.

How would this synthesis of patterns into an RA be realized in practice? As the synthesis is highly dependent on the high-priority requirements of the RA stakeholders, it is impossible to prescribe a one-size-fits-all architecture. Instead, we describe the procedure of how an RA would be synthesized for a specific use case, illustrated in Figure 4. With the prioritized QAs (from Step 1) as a guide, components (from Step 2) that address these attributes (using the analysis from Step 3) will be identified and integrated into the architecture. Practically, an initial design meeting would be scheduled to review the existing workflows and patterns concerning the requirements. During this meeting, a candidate RA will be crafted, following high-level architectural principles

² <https://serpapi.com/google-scholar-api>, accessed: 2024-01-05.



■ **Figure 4** An example of RA synthesis. Stakeholders have identified a set of QAs (scalability, domain-specificity, and extensibility) and a specific use case (graph extension of an enterprise KG). The team of architects has taken this as input, selected an adequate pattern F2 (fusion 2) based on its support for the indicated QAs and use case, and synthesized a proposed RA that uses a trained subject classifier to perform graph extension based on the KG and data from a content repository. After a process of iterative refinement, choices are made about specific technologies to use, and a concrete RA is proposed.

and approaches. A core team of architects is then essential to conduct a collaborative design session. In this session, the RA's architecture, its components, and their interactions are laid out, creating a platform for real-time discussions and potential modifications. During these discussions, the SWeMLS knowledge graph can provide information about potential alternative technologies for the components. To refine the design further, a series of subsequent sessions can be organized that invite a broader set of participants. The feedback gathered from these sessions can be used to iterate and enhance the design.

Following a similar procedure, an example RA for KE in the domain of newsrooms and journalism has been provided by [61]. A critical future work is to apply our process to other use cases with potentially different requirements.

4.4.2 Hypothetical scenario

We proceed to illustrate this process with a hypothetical **scenario** (Figure 4). Business stockholders at a large enterprise have identified that there is a need to improve the discoverability of content on a corporate website. The company has a repository of content, including product information, articles, blog posts, case studies, and user guides. However, users often struggle to find the content they need, leading to frustration, reduced engagement, and potentially lost sales opportunities. The company recognizes that better content recommendations and more intuitive navigation can significantly enhance the user experience and drive business outcomes.

5:16 Reference Architectures for Knowledge Engineering

Based on their understanding of industry best practices, the stakeholders determine that a solution that performs subject tagging of content using a domain-specific ontology will support the discoverability of content for tasks their users are attempting to accomplish. They identify several **quality attributes** that they want to ensure such a solution addresses:

- Scalability: The solution should handle a large and growing volume of content and user interactions.
- Domain-specificity: The solution should provide subject tagging from a domain-specific taxonomy of vocabulary terms and definitions.
- Extensibility: The solution should be extensible as new content and subjects become available.

These QAs, along with additional FRs, are presented to a team of architects. The architects review the **design patterns** captured in the SWeMLS KG to determine which of the identified design patterns most closely address these QAs and requirements. The knowledge graph supports the review process by surfacing relevant papers, case studies, and benchmarks for similar systems and use cases. Querying the KG, the architects identify the F2 design pattern as describing systems that match the stakeholder QAs and the use case. Both of the systems corresponding to F2 are similar to the stakeholder use case, and both provide evidence that the pattern can address the specified QAs. Based on this review, the **F2** design pattern is selected.

The architects then specify how the requirements and use case can be addressed by instantiating the components in the F2 design pattern into a **proposed RA**, as follows:

- The input symbolic representation (s) is an enterprise KG that captures the semantics of the content repository and application domain, including the domain subject taxonomy. The KG should capture key entities like products, articles, and customer segments, along with their relationships.
- The input data (d) is the content on the corporate website.
- The model (M) component is a combination of ML and NLP technologies that given the KG and content, classifies the content according to the domain-specific subject taxonomy.
- The output symbolic representation (s) are relations to be added to the knowledge graph to link content on the website to relevant concepts in the vocabulary.

Given these decisions, the architects then proceed to make additional choices for what specific technologies are to be used to implement each component into a **concrete RA**:

- Knowledge graph (s): this could be stored in a labeled property graph database, an RDF triple store, or a relational database with a graph-friendly schema.
- Content repository (d): given the existing website, the input data may reside in various enterprise systems like content management systems or customer-customer relationship management systems.
- Model (M): the model is responsible for producing multi-class subject classifications from the input data and KG, and updating the KG with this new knowledge. Suitable approaches could include hybrid models that combine text, user interactions, and graph structure, e.g., using transformer architectures like BERT or pre-trained language models like GPT-4.

The architects additionally consider factors such as the volume and variety of input data, the complexity of the topic taxonomy, explainability requirements, and the team's AI/ML skills in making their decisions about how to instantiate the RA, including the following considerations:

- The iterative nature of the F2 pattern, where the output enhances the KG, can support continuous improvement of recommendations as new content is incorporated.
- The use of a KG as the core representation to aid explainability, as the relationships between content and topics can be traced and visualized, potentially helping content managers optimize the content strategy and troubleshoot issues.

- The separation of concerns in the F2 design pattern, with dedicated components for data ingestion, model training, and KG management, promotes scalability and performance, as each component can be independently optimized and scaled based on the workload.

The architects then go through a final process of determining the **final proposed architecture**, potentially including:

- Assessing the current state of the KG and identifying gaps in topic coverage
- Inventorying available data sources and evaluating their relevance and quality
- Experimenting with different modeling approaches and comparing their accuracy, scalability, and interpretability
- Validate model outputs with subject matter experts and through user testing

The final proposed architecture is then documented to support the evaluation process described in the next phase of the process. Once the reference architecture has been defined, it can be **stored** in the SWeMLS KG. This allows the RA to be shared and reused in other content classification applications within the enterprise. Some examples of how elements of the reference architecture definition can be mapped into the SWeMLS knowledge graph are:

- The overall RA for content recommendation can be represented as an instance of the `swemls:System` class. The specific pattern it implements (F2) can be indicated using the `swemls:hasCorrespondingPattern` property.
- The business problem of improving content discoverability on the corporate website can be described using the `swemls:Task` class.
- The various data sources used to build and enhance the KG, such as content metadata, user interaction logs, and external taxonomies, can be captured using the `swemls:Data` class.
- The KG serving as the core symbolic representation can be modeled as an instance of the `swemls:SemanticWebResource` class. The specific KG technology used can be specified using the `swTechnology` property.
- The machine learning model used to learn topic classifications can be represented using the `swemls:Model` class.
- The process of training the machine learning models using the input data and KG can be represented using the `swemls:ProcessingEngine` class.
- The specific tools, libraries, and frameworks used to implement the RA components can be captured as instances of the relevant classes, including `swemls:Data`, `swemls:Model`, and `swemls:SemanticWebResource`.

By mapping the RA to the SWeMLS ontology in this way, we establish a structured and semantically rich representation of the architectural knowledge that can be a resource in the evaluation process described in the next section. The ontology classes, properties, and relationships provide a standardized vocabulary to describe the various aspects of the RA, from the business goals and QAs to the technical components and best practices. This consistent representation facilitates comparison, integration, and reasoning across different RAs and domain applications.

4.5 RA evaluation through instantiation and use

Once specified, architectures synthesized in this manner from design patterns can then be evaluated through a lightweight version of the Architecture Tradeoff Analysis Method (ATAM). ATAM [47] is a risk-mitigation process used to identify architectural risks that have implications in fulfilling quality attributes. As originally proposed by CMU's Software Engineering Institute, this process involved a multi-day face-to-face gathering of stakeholders and architects. A lightweight ATAM process [73] is a streamlined version of the traditional ATAM, focusing on a shorter,

more rapid timeframe and often less resource-intensive evaluation of architectural decisions. The use of web-conferencing and real-time collaborative document editors allows this process to be conducted remotely, increasing the ability to gather a large and diverse group of stakeholders. First, we will identify and recruit a set of stakeholders for an ATAM session. On the day of the session, the stakeholders will be presented with an agenda for the session that defines the scope of the evaluation and presents the RA, identifying architectural approaches used. This will be followed by a presentation of user scenarios relative to evaluating the RA. The user scenarios for knowledge creation and maintenance processes should include capabilities for data integration from multiple structured sources [20], data quality checks [67], entity resolution [17], ontology merging and alignment [15, 62], query optimization [37], and natural language processing [76] (cf. Table 1). Moreover, the production processes should be automated to enable efficient updates and maintenance of the knowledge artifact [59]. In cases where the KE involves the use of personally identifiable information or other sensitive data for knowledge elicitation or training ML components, there is the danger of leakage of sensitive information; in addition, ML components themselves can inadvertently leak data under adversarial attack [60]. Therefore, the production process should incorporate mechanisms for security and privacy, as well as access control mechanisms to ensure that the data stays secure and that only authorized users have access. It is worth observing that many of these issues have been explored to date in the more generic context of data engineering and data science architectures and platforms. Once the stakeholders have considered the various scenarios, they can proceed to collaboratively analyze the scenarios, identifying risks and trade-offs, and gathering feedback, focusing on potential refinements and architectural alternatives. The stakeholders will then document risks, trade-offs, architectural decisions, and the reasons for them, finishing by summarizing the final consensus RA.

4.6 RA instantiation in a concrete software architecture

Given a consensus RA, we can proceed to finalize a comprehensive architectural blueprint. The RA does not provide absolute recommendations on such choices, assuming that those are stakeholder need-dependent. It does, however, prescribe an association between different requirements, architectural patterns, and adequate implementations. For each component, the range of options for its instantiation using existing software packages or through bespoke development will be identified. Here, we are inspired by recent toolkits for knowledge graphs, like KGTK [43], which connect knowledge engineering operations by defining a universal interface format and abstracting the implementation of each component from the user. The implementation of each component relies on thorough research and consideration of the best existing tool or implementation that can be wrapped, i.e., that the software can provide an interface to. For instance, one could provide an interface to Pytorch-Biggraph [52] for knowledge completion, Shape expressions (Shex) tools [88] for using constraints to evaluate quality, and RLTK [97] for record linkage across knowledge artifacts. However, as KGTK and similar toolkits are built based on an implicit set of use cases and user requirements, further investigation is required to assess whether they will align with emerging architectural contributions like the set of boxology patterns.

A strategic approach to instantiating an RA involves a phased implementation. Each phase should predominantly focus on one specific component. During this development phase, constant testing and evaluation of each component will be performed to ensure the component aligns with the predefined QAs and specific scenarios. After the conclusion of each phase, feedback will be gathered from all involved stakeholders. This iterative process will ensure the architecture remains relevant and effective, as necessary revisions based on the feedback can be made. The resulting implementation will be open-sourced and thoroughly documented in a publicly accessible code repository. After the entire process is complete, the system's efficacy will be tested in pilot

trials facilitated by the stakeholders. During these trials, relevant data on system performance and quality assessment will be collected to ensure the architecture's robustness and efficiency. Notably, the implemented RA would serve as a comprehensive framework that enables decisions on technology for representation, integration, and quality assurance, among others, to be made based on high-priority requirements. While the implementation is meant to be prescriptive and enable efficient KE by profiles with various backgrounds, goals, and levels of expertise, we acknowledge that these recommendations should be considered relative to the stakeholders' needs.

5 Conclusions

Knowledge engineering, as a process of creating and maintaining knowledge artifacts, has remained relevant throughout the history of AI. In light of the heterogeneous requirements and KE use cases, on the one hand, and the emergence of architectural components and partial workflows, on the other hand, this paper makes a case for developing reference architectures for KE. Following software engineering practices, an RA would provide an organizational principle for isolated systemic patterns, thus providing a key contribution to this ongoing work that enables the knowledge engineering field to be systematized. A reference architecture consolidates the patterns while simultaneously considering its scope, defined through a set of use cases and their corresponding requirements, distilled as quality attributes and functional requirements. The synthesis of the architecture is an iterative process, inspired by success stories of reference architectures for service-oriented design, e-government, and the automotive sector. A key aspect of the development is its evaluation through instantiation and use with representative users for representative KE tasks. As a final step, the reference architecture components need to be instantiated into software, thus closing the cycle between user needs and existing technological capabilities.

While this paper outlines a roadmap for devising comprehensive and requirement-grounded RAs for KE, its realization in practice is partial at present. We present a broad definition of scope through a definition of representative tasks and distillation of 23 quality attributes and 8 functional requirements, which could be narrowed down given a specific use case. We take the recently identified collection of system patterns for neurosymbolic KE as information sources, providing initial components that can be used to construct an RA. We present an architectural analysis, as a direct mapping between QAs and the identified architectural patterns, detecting requirements with various levels of support. The steps of synthesizing an RA from patterns, evaluating the RA through instantiation and use, and instantiating the RA into software are presented as prescriptive, consolidating best practices and methodologies from software engineering through step-by-step processes, because these steps are highly dependent on the specific use cases. Each of these steps requires the dedicated effort of iterative design, development, implementation, and evaluation of an RA, which we plan to pursue as the next steps for representative subsets of tasks and domains. We believe that the presented methodology for devising RAs for KE provides an important extension of emerging work that systematizes KE methods, by providing a mechanism to associate architectural patterns with user requirements and identify potential gaps. We invite the broader community of interested researchers and developers to join us in these discussions and complement our future efforts in consolidating KE practices.

References

- 1 David Abián, Albert Meroño-Peñuela, and Elena Simperl. An analysis of content gaps versus user needs in the wikidata knowledge graph. In *International Semantic Web Conference*, pages 354–374. Springer, 2022. doi:10.1007/978-3-031-19433-7_21.
- 2 Badr AlKhamissi, Millicent Li, Asli Celikyilmaz, Mona Diab, and Marjan Ghazvininejad. A review on language models as knowledge bases. *arXiv*

- preprint *arXiv:2204.06031*, 2022. doi:10.48550/arXiv.2204.06031.
- 3 Bradley P. Allen, Filip Ilievski, and Saurav Joshi. Identifying and consolidating knowledge engineering requirements. *arXiv preprint arXiv:2306.15124*, 2023. doi:10.48550/arXiv.2306.15124.
 - 4 Jürgen Angele, Dieter Fensel, Dieter Landes, and Rudi Studer. Developing knowledge-based systems with mike. *domain modelling for interactive systems design*, pages 9–38, 1998.
 - 5 Samuil Angelov, Paul Grefen, and Danny Greefhorst. A classification of software reference architectures: Analyzing their success and effectiveness. In *2009 Joint Working IEEE/IFIP Conference on Software Architecture & European Conference on Software Architecture*, pages 141–150. IEEE, 2009. doi:10.1109/WICSA.2009.5290800.
 - 6 Ali Arsanjani, Liang-Jie Zhang, Michael Ellis, Abdul Allam, and Kishore Channabasavaiah. S3: A service-oriented reference architecture. *IT professional*, 9(3):10–17, 2007. doi:10.1109/MITP.2007.53.
 - 7 Pouya Ataei and Alan Litchfield. The state of big data reference architectures: A systematic literature review. *IEEE Access*, 2022. doi:10.1109/ACCESS.2022.3217557.
 - 8 Len Bass, Paul Clements, and Rick Kazman. *Software architecture in practice*. SEI Series in Software Engineering. Addison-Wesley Professional, fourth edition, 2022.
 - 9 Wouter Beek, Laurens Rietveld, Stefan Schlobach, and Frank van Harmelen. Lod laundromat: Why the semantic web needs centralization (even if we don't like it). *IEEE Internet Computing*, 20(2):78–81, 2016. doi:10.1109/MIC.2016.43.
 - 10 Emily M. Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. On the dangers of stochastic parrots: Can language models be too big? In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '21, pages 610–623, New York, NY, USA, 2021. Association for Computing Machinery. doi:10.1145/3442188.3445922.
 - 11 Tim Berners-Lee, James Hendler, and Ora Lassila. The semantic web. *Scientific american*, 284(5):34–43, 2001.
 - 12 Juergen Boltdt. The common object request broker: Architecture and specification. Specification formal/97-02-25, Object Management Group, jul 1995. URL: <http://www.omg.org/cgi-bin/doc?formal/97-02-25>.
 - 13 Matt Bornstein, Jennifer Li, and Casado. Martin. Emerging architectures for modern data infrastructure, 2020. <https://future.com/emerging-architectures-modern-data-infrastructure/>.
 - 14 Anna Breit, Laura Waltersdorfer, Fajar J Ekaputra, Marta Sabou, Andreas Ekelhart, Andreea Iana, Heiko Paulheim, Jan Portisch, Artem Revenko, Annette ten Teije, et al. Combining machine learning and semantic web: A systematic mapping study. *ACM Computing Surveys*, 2023. doi:10.1145/3586163.
 - 15 Niladri Chatterjee, Neha Kaushik, Deepali Gupta, and Ramneek Bhatia. Ontology merging: A practical perspective. In *Information and Communication Technology for Intelligent Systems (ICTIS 2017)-Volume 2*, pages 136–145. Springer, 2018.
 - 16 Surajit Chaudhuri and Umeshwar Dayal. An overview of data warehousing and olap technology. *ACM Sigmod record*, 26(1):65–74, 1997. doi:10.1145/248603.248616.
 - 17 Vassilis Christophides, Vasilis Efthymiou, Themis Palpanas, George Papadakis, and Kostas Stefanidis. An overview of end-to-end entity resolution for big data. *ACM Computing Surveys (CSUR)*, 53(6):1–42, 2020. doi:10.1145/3418896.
 - 18 Robert Cloutier, Gerrit Muller, Dinesh Verma, Roshanak Nilchiani, Eirik Hole, and Mary Bone. The concept of reference architectures. *Systems Engineering*, 13(1):14–27, 2010. doi:10.1002/sys.20129.
 - 19 Xin Luna Dong and Divesh Srivastava. Schema alignment. In *Big Data Integration*, pages 31–61. Springer, 2015.
 - 20 Fajar Ekaputra, Marta Sabou, Estefanía Serral Asensio, Elmar Kiesling, and Stefan Biffl. Ontology-based data integration in multidisciplinary engineering environments: A review. *Open Journal of Information Systems*, 4(1):1–26, 2017. URL: https://www.ronpub.com/ojais/OJIS_2017v4i1n01_Ekaputra.html.
 - 21 Fajar J Ekaputra, Majlinda Llugiqi, Marta Sabou, Andreas Ekelhart, Heiko Paulheim, Anna Breit, Artem Revenko, Laura Waltersdorfer, Kheir Eddine Farfar, and Sören Auer. Describing and organizing semantic web and machine learning systems in the swemls-kg. In *European Semantic Web Conference*, pages 372–389. Springer, 2023. doi:10.1007/978-3-031-33455-9_22.
 - 22 Julian Ereth. Dataops-towards a definition. *LWDA*, 2191:104–112, 2018. URL: <https://ceur-ws.org/Vol-2191/paper13.pdf>.
 - 23 Vadim Ermolayev, Sotiris Batsakis, Natalya Keberle, Olga Tatarintseva, and Grigoris Antoniou. Ontologies of time: Review and trends. *International Journal of Computer Science & Applications*, 11(3), 2014. URL: <http://www.tmrfindia.org/ijcsa/v11i34.pdf>.
 - 24 Edward A Feigenbaum. The art of artificial intelligence: Themes and case studies of knowledge engineering. In *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, volume 2. Boston, 1977. URL: <http://ijcai.org/Proceedings/77-2/Papers/092.pdf>.
 - 25 Edward A. Feigenbaum. A personal view of expert systems: Looking back and looking ahead. *Expert Systems with Applications*, 5(3):193–201, 1992. Special Issue: The World Congress on Expert System. doi:10.1016/0957-4174(92)90004-C.
 - 26 Mariano Fernández-López, Asuncion Gomez-Perez, and Natalia Juristo. Methontology: from ontological art towards ontological engineering. *Engineering Workshop on Ontological Engineering (AAAI97)*, mar 1997.
 - 27 DRAFT NIST Big Data Interoperability Framework. Draft nist big data interoperability framework: Volume 6, reference architecture. *NIST Special Publication*, 1500:6, 2015.

- 28 Aldo Gangemi and Valentina Presutti. Ontology design patterns. In *Handbook on ontologies*, pages 221–243. Springer, 2009. doi:10.1007/978-3-540-92673-3_10.
- 29 Lina Garcés, Silverio Martínez-Fernández, Lucas Oliveira, Pedro Valle, Claudia Ayala, Xavier Franch, and Elisa Yumi Nakagawa. Three decades of software reference architectures: A systematic mapping study. *Journal of Systems and Software*, 179:111004, 2021. doi:10.1016/j.jss.2021.111004.
- 30 John H Gennari, Mark A Musen, Ray W Ferguson, William E Grosso, Monica Crubézy, Henrik Eriksson, Natalya F Noy, and Samson W Tu. The evolution of protégé: an environment for knowledge-based systems development. *International Journal of Human-computer studies*, 58(1):89–123, 2003. doi:10.1016/S1071-5819(02)00127-1.
- 31 Randy Goebel, Sandra Zilles, Christoph Ringlstetter, Andreas Dengel, and Gunnar Aastrand Grimnes. What is the role of the semantic layer cake for guiding the use of knowledge representation and machine learning in the development of the semantic web? In *AAAI Spring Symposium: Symbiotic Relationships between Semantic Web and Knowledge Engineering*, pages 45–50, 2008. URL: <http://www.aaai.org/Library/Symposia/Spring/2008/ss08-07-006.php>.
- 32 Asunción Gómez-Pérez, Mariano Fernández-López, and Oscar Corcho. *Ontological Engineering: with examples from the areas of Knowledge Management, e-Commerce and the Semantic Web*. Springer Science & Business Media, 2006.
- 33 Paul Groth, Elena Simperl, Marieke van Erp, and Denny Vrandečić. Knowledge graphs and their role in the knowledge engineering of the 21st century (dagstuhl seminar 22372). *Dagstuhl Reports*, 12(9), 2023. doi:10.4230/DagRep.12.9.60.
- 34 Lin Guan, Karthik Valmeekam, Sarath Sreedharan, and Subbarao Kambhampati. Leveraging pre-trained large language models to construct and utilize world models for model-based task planning. *Advances in Neural Information Processing Systems*, 36:79081–79094, 2023. URL: http://papers.nips.cc/paper_files/paper/2023/hash/f9f54762cbb4fe4dbffdd4f792c31221-Abstract-Conference.html.
- 35 Anaïs Guillem, Antoine Gros, Kévin Réby, Violette Abergel, and Livio De Luca. Rcc8 for cidoc crm: semantic modeling of mereological and topological spatial relations in notre-dame de paris. In *SWODCH'23: International Workshop on Semantic Web and Ontology Design for Cultural Heritage*, 2023. URL: <https://ceur-ws.org/Vol-3540/paper2.pdf>.
- 36 Olaf Hartig. Reflections on Linked Data Querying and other Related Topics. <https://olafhartig.de/slides/Slides-DKG-SWSA-Talk.pdf>, 2022. Accessed: 2022-03-17.
- 37 Olaf Hartig, Christian Bizer, and Johann-Christoph Freytag. Executing sparql queries over the web of linked data. In *The Semantic Web-ISWC 2009: 8th International Semantic Web Conference, ISWC 2009, Chantilly, VA, USA, October 25-29, 2009. Proceedings 8*, pages 293–309. Springer, 2009. doi:10.1007/978-3-642-04930-9_19.
- 38 Frederick Hayes-Roth, Donald A Waterman, and Douglas B Lenat. *Building expert systems*. Addison-Wesley Longman Publishing Co., Inc., 1983.
- 39 James A Hendler. Tonight’s dessert: Semantic web layer cakes. In *European Semantic Web Conference*, pages 1–1. Springer, 2009. doi:10.1007/978-3-642-02121-3_1.
- 40 Aidan Hogan. The semantic web: Two decades on. *Semantic Web*, 11(1):169–185, 2020. doi:10.3233/SW-190387.
- 41 Carlos A Iglesias, Mercedes Garijo, José C González, and Juan R Velasco. Analysis and design of multiagent systems using mas-commonkads. In *Intelligent Agents IV Agent Theories, Architectures, and Languages: 4th International Workshop, ATAL'97 Providence, Rhode Island, USA, July 24–26, 1997 Proceedings 4*, pages 313–327. Springer, 1998. doi:10.1007/BFb0026768.
- 42 Ana Iglesias-Molina, Kian Ahrabian, Filip Ilievski, Jay Pujara, and Oscar Corcho. Comparison of knowledge graph representations for user consumption scenarios. In *International Semantic Web Conference (ISWC) Research Track*, 2023. doi:10.1007/978-3-031-47240-4_15.
- 43 Filip Ilievski, Daniel Garijo, Hans Chalupsky, Naren Teja Divvala, Yixiang Yao, Craig Rogers, Rongpeng Li, Jun Liu, Amandeep Singh, Daniel Schwabe, and Pedro Szekely. Kgtk: a toolkit for large knowledge graph manipulation and analysis. In *International Semantic Web Conference*, pages 278–293. Springer, 2020. doi:10.1007/978-3-030-62466-8_18.
- 44 Filip Ilievski, Pedro Szekely, and Bin Zhang. Cskg: The commonsense knowledge graph. In *Extended Semantic Web Conference (ESWC)*, 2021. doi:10.1007/978-3-030-77385-4_41.
- 45 Prateek Jain, Pascal Hitzler, Amit P Sheth, Kunal Verma, and Peter Z Yeh. Ontology alignment for linked open data. In *International semantic web conference*, pages 402–417. Springer, 2010. doi:10.1007/978-3-642-17746-0_26.
- 46 Henry Kautz. The third ai summer: Aaai robert s. engelmore memorial lecture. *AI Magazine*, 43(1):93–104, 2022. doi:10.1609/aimag.v43i1.19122.
- 47 Rick Kazman, Mark Klein, and Paul Clements. *ATAM: Method for architecture evaluation*. Carnegie Mellon University, Software Engineering Institute Pittsburgh, PA, 2000.
- 48 Elisa F Kendall and Deborah L McGuinness. *Ontology engineering*. Morgan & Claypool Publishers, 2019. doi:10.2200/S00834ED1V01Y201802WBE018.
- 49 Vijay Khatri and Carol V Brown. Designing data governance. *Communications of the ACM*, 53(1):148–152, 2010. doi:10.1145/1629175.1629210.
- 50 Gongjin Lan, Ting Liu, Xu Wang, Xueli Pan, and Zhisheng Huang. A semantic web technology index. *Scientific reports*, 12(1):3672, 2022.
- 51 Doug Lenat and Gary Marcus. Getting from generative ai to trustworthy ai: What llms might learn from cyc. *arXiv preprint arXiv:2308.04445*, 2023. doi:10.48550/arXiv.2308.04445.

- 52 Adam Lerer, Ledell Wu, Jiajun Shen, Timothee Lacroix, Luca Wehrstedt, Abhijit Bose, and Alex Peysakhovich. Pytorch-biggraph: A large scale graph embedding system. *Proceedings of Machine Learning and Systems*, 1:120–131, 2019. URL: <https://proceedings.mlsys.org/book/282.pdf>.
- 53 Bo Li, Peng Qi, Bo Liu, Shuai Di, Jingen Liu, Jiquan Pei, Jinfeng Yi, and Bowen Zhou. Trustworthy ai: From principles to practices. *ACM Computing Surveys*, 55(9):1–46, 2023. doi:10.1145/3555803.
- 54 Sebastian Lobentanzer, Patrick Aloy, Jan Baumbach, Balazs Bohar, Vincent J Carey, Pornpimol Charoentong, Katharina Danhauser, Tunca Doğan, Johann Dreo, Ian Dunham, et al. Democratizing knowledge representation with biocypher. *Nature Biotechnology*, pages 1–4, 2023.
- 55 Sebastian Lobentanzer, Patrick Aloy, Jan Baumbach, Balazs Bohar, Vincent J Carey, Pornpimol Charoentong, Katharina Danhauser, Tunca Doğan, Johann Dreo, Ian Dunham, et al. Democratizing knowledge representation with biocypher. *Nature Biotechnology*, 41(8):1056–1059, 2023.
- 56 Elisa Y Nakagawa, Fabiano C Ferrari, Mariela MF Sasaki, and José C Maldonado. An aspect-oriented reference architecture for software engineering environments. *Journal of Systems and Software*, 84(10):1670–1684, 2011. doi:10.1016/j.jss.2011.04.052.
- 57 Allen Newell, John Calman Shaw, and Herbert A Simon. Elements of a theory of human problem solving. *Psychological review*, 65(3):151, 1958.
- 58 Natalya F Noy, Deborah L McGuinness, et al. Ontology development 101: A guide to creating your first ontology, 2001.
- 59 Natasha Noy, Yuqing Gao, Anshu Jain, Anant Narayanan, Alan Patterson, and Jamie Taylor. Industry-scale knowledge graphs: Lessons and challenges: Five diverse technology companies show how it's done. *Queue*, 17(2):48–75, 2019. doi:10.1145/3329781.3332266.
- 60 Marc Gallofré Ocaña, Tareq Al-Moslmi, and A. Opdahl. Data privacy in journalistic knowledge platforms. In *International Conference on Information and Knowledge Management*, 2020. URL: <https://ceur-ws.org/Vol1-2699/paper44.pdf>.
- 61 Marc Gallofré Ocaña and Andreas L Opdahl. A software reference architecture for journalistic knowledge platforms. *Knowledge-Based Systems*, 276:110750, 2023. doi:10.1016/j.knsys.2023.110750.
- 62 Lorena Otero-Cerdeira, Francisco J Rodríguez-Martínez, and Alma Gómez-Rodríguez. Ontology matching: A literature review. *Expert Systems with Applications*, 42(2):949–971, 2015. doi:10.1016/j.eswa.2014.08.032.
- 63 Heiko Paulheim. Knowledge graph refinement: A survey of approaches and evaluation methods. *Semantic web*, 8(3):489–508, 2017. doi:10.3233/SW-160218.
- 64 Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. Language models as knowledge bases? In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan, editors, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2463–2473, Hong Kong, China, nov 2019. Association for Computational Linguistics. doi:10.18653/v1/D19-1250.
- 65 Alessandro Piscopo and Elena Simperl. Who models the world? collaborative ontology creation and user roles in wikidata. *Proceedings of the ACM on Human-Computer Interaction*, 2(CSCW):1–18, 2018. doi:10.1145/3274410.
- 66 María Poveda-Villalón, Alba Fernández-Izquierdo, Mariano Fernández-López, and Raúl García-Castro. Lot: An industrial oriented ontology engineering framework. *Engineering Applications of Artificial Intelligence*, 111:104755, 2022. doi:10.1016/j.engappai.2022.104755.
- 67 Alun Preece. Evaluating verification and validation methods in knowledge engineering. In *Industrial knowledge management: A micro-level approach*, pages 91–104. Springer, 2001.
- 68 Jason Priem, Heather Piwowar, and Richard Orr. Openalex: A fully-open index of scholarly works, authors, venues, institutions, and concepts. *arXiv preprint arXiv:2205.01833*, 2022. doi:10.48550/arXiv.2205.01833.
- 69 Qinjun Qiu, Zhong Xie, Liang Wu, and Liufeng Tao. Dictionary-based automated information extraction from geological documents using a deep learning algorithm. *Earth and Space Science*, 7(3):e2019EA000993, 2020.
- 70 F.P. Ramsey. Knowledge. In *F.P. Ramsey: Philosophical Papers*, pages 110–111. Cambridge University Press, 1929.
- 71 Franck Ravat and Yan Zhao. Data lakes: Trends and perspectives. In *Database and Expert Systems Applications: 30th International Conference, DEXA 2019, Linz, Austria, August 26–29, 2019, Proceedings, Part I 30*, pages 304–313. Springer, 2019. doi:10.1007/978-3-030-27615-7_23.
- 72 Marta Sabou, Majlinda Llugiqi, Fajar J Ekaputra, Laura Waltersdorfer, and Stefani Tsaneva. Knowledge engineering in the age of neurosymbolic systems. *Neurosymbolic AI Journal (under review)*, 2024.
- 73 Mahdi Sahlabadi, Ravie Chandren Muniyandi, Zarina Shukur, and Faizan Qamar. Lightweight software architecture evaluation for industry: A comprehensive review. *Sensors*, 22(3):1252, 2022. doi:10.3390/s22031252.
- 74 Salman Salloum, Ruslan Dautov, Xiaojun Chen, Patrick Xiaogang Peng, and Joshua Zhexue Huang. Big data analytics on apache spark. *International Journal of Data Science and Analytics*, 1:145–164, 2016. doi:10.1007/s41060-016-0027-9.
- 75 Michael Schade. How ChatGPT and Our Language Models Are Developed, 2023. URL: <https://help.openai.com/en/articles/7842364-how-chatgpt-and-our-language-models-are-developed>.
- 76 Phillip Schneider, Tim Schopf, Juraj Vladika, Mikhail Galkin, Elena Paslaru Bontas Simperl, and Florian Matthes. A decade of knowledge graphs in natural language processing: A survey. In *ACL*, 2022. URL: <https://api.semanticscholar.org/CorpusID:252683270>.

- 77 August Th Schreiber, Guus Schreiber, Hans Akkermans, Anjo Anjewierden, Nigel Shadbolt, Robert de Hoog, Walter Van de Velde, and Bob Wielinga. *Knowledge engineering and management: the CommonKADS methodology*. MIT press, 2000.
- 78 Kartik Shenoy, Filip Ilievski, Daniel Garijo, Daniel Schwabe, and Pedro Szekely. A study of the quality of wikidata. *Journal of Web Semantics*, 2021. doi:10.1016/j.websem.2021.100679.
- 79 Umutcan Simsek, Elias Kärle, Kevin Angele, Elwin Huaman, Juliette Opendenplatz, Dennis Sommer, Jürgen Umbrich, and Dieter Fensel. A knowledge graph perspective on knowledge engineering. *SN Computer Science*, 4(1):16, 2022. doi:10.1007/s42979-022-01429-x.
- 80 Dezhao Song, Frank Schilder, Shai Hertz, Giuseppe Saltini, Charese Smiley, Phani Nivarthi, Oren Hazai, Dudi Landau, Mike Zaharkin, Tom Zielund, et al. Building and querying an enterprise knowledge graph. *IEEE Transactions on Services Computing*, 12(3):356–369, 2017. doi:10.1109/TSC.2017.2711600.
- 81 Mirosław Staron and Mirosław Staron. Autosar (automotive open system architecture). *Automotive Software Architectures: An Introduction*, pages 97–136, 2021.
- 82 Monika Steidl, Michael Felderer, and Rudolf Ramler. The pipeline for the continuous development of artificial intelligence models—current state of research and practice. *Journal of Systems and Software*, 199:111615, 2023. doi:10.1016/j.jss.2023.111615.
- 83 Mari Carmen Suárez-Figueroa, Asunción Gómez-Pérez, and Mariano Fernández-López. The neon methodology for ontology engineering. In *Ontology engineering in a networked world*, pages 9–34. Springer, 2011. doi:10.1007/978-3-642-24794-1_2.
- 84 Gytė Tamašauskaitė and Paul Groth. Defining a knowledge graph development process through a systematic review. *ACM Transactions on Software Engineering and Methodology*, 2022. doi:10.1145/3522586.
- 85 Richard N Taylor, Nenad Medvidović, and Eric M Dashofy. *Software architecture: foundations, theory, and practice*. John Wiley & Sons, Inc., 2010.
- 86 WDQS Search Team. WDQS Backend Alternatives: The Process, Details and Results. https://www.wikidata.org/wiki/File:WDQS_Backend_Alternatives_working_paper.pdf, 2022. Accessed: 2022-08-15.
- 87 Karim Tharani. Much more than a mere technology: A systematic review of wikidata in libraries. *The Journal of Academic Librarianship*, 47(2):102326, 2021.
- 88 Katherine Thornton, Harold Solbrig, Gregory S Stupp, Jose Emilio Labra Gayo, Daniel Mietchen, Eric Prud’Hommeaux, and Andra Waagmeester. Using shape expressions (shex) to share rdf data models and to guide curation with rigorous validation. In *The Semantic Web: 16th International Conference, ESWC 2019, Portorož, Slovenia, June 2–6, 2019, Proceedings 16*, pages 606–620. Springer, 2019. doi:10.1007/978-3-030-21348-0_39.
- 89 Ilaria Tiddi, Victor De Boer, Stefan Schlobach, and André Meyer-Vitali. Knowledge engineering for hybrid intelligence. In *Proceedings of the 12th Knowledge Capture Conference 2023*, pages 75–82, 2023. doi:10.1145/3587259.3627541.
- 90 Riccardo Tommasini, Filip Ilievski, and Thilini Wijesiriwardene. The internet meme knowledge graph. In *ESWC*, 2023. doi:10.1007/978-3-031-33455-9_21.
- 91 Michael van Bekkum, Maaïke de Boer, Frank van Harmelen, André Meyer-Vitali, and Annette ten Teije. Modular design patterns for hybrid learning and reasoning systems: a taxonomy, patterns and use cases. *Applied Intelligence*, 51(9):6528–6546, 2021. doi:10.1007/s10489-021-02394-3.
- 92 Frank Van Harmelen and Annette Ten Teije. A boxology of design patterns for hybrid learning and reasoning systems. *Journal of Web Engineering*, 18(1-3):97–123, 2019. doi:10.13052/jwe1540-9589.18133.
- 93 Laura Waltersdorfer, Anna Breit, Fajar J Ekaputra, Marta Sabou, Andreas Ekelhart, Andreea Iana, Heiko Paulheim, Jan Portisch, Artem Revenko, Annette ten Teije, et al. Semantic web machine learning systems: An analysis of system patterns. In *Compendium of Neurosymbolic Artificial Intelligence*, pages 77–99. IOS Press, 2023. doi:10.3233/FAIA230136.
- 94 Bob J Wielinga, A Th Schreiber, and Jost A Breuker. Kads: A modelling approach to knowledge engineering. *Knowledge acquisition*, 4(1):5–53, 1992.
- 95 Hans Friedrich Witschel, Charuta Pande, Andreas Martin, Emanuele Laurenzi, and Knut Hinkelmann. Visualization of patterns for hybrid learning and reasoning with human involvement. In *New Trends in Business Information Systems and Technology: Digital Innovation and Digital Business Transformation*, pages 193–204. Springer, 2020.
- 96 Dong Yang, Lixin Tong, Yan Ye, and Hongwei Wu. Applying commonkads and semantic web technologies to ontology-based e-government knowledge systems. In *The Semantic Web—ASWC 2006: First Asian Semantic Web Conference, Beijing, China, September 3-7, 2006. Proceedings 1*, pages 336–342. Springer, 2006. doi:10.1007/11836025_34.
- 97 Yixiang Yao, Pedro Szekely, and Jay Pujara. Extensible and scalable entity resolution for financial datasets using rltk. In *Proceedings of the 5th Workshop on Data Science for Macro-modeling with Financial and Economic Datasets*, pages 1–1, 2019. doi:10.1145/3336499.3338008.
- 98 Nasser Zalmout, Chenwei Zhang, Xian Li, Yan Liang, and Xin Luna Dong. All you need to know to build a product knowledge graph. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 4090–4091, 2021. doi:10.1145/3447548.3470825.

