Department of Computing and Software

Faculty of Engineering — McMaster University

A Data-Centered Framework for Domain Knowledge Representation

by

Alicia Marinache, Ridha Khedri, and Wendy MacCaull

CAS Report Series **CAS-19-09-RK**

Department of Computing and Software **September 2019**
Information Technology Building
McMaster University
1280 Main Street West Hamilton, Ontario, Canada L8S 4K1

A Data-Centered Framework for Domain Knowledge Representation

Alicia Marinache[1], Ridha Khedri[1], and Wendy MacCaull[2]

[1] Department of Computing and Software, Faculty of Engineering,
McMaster University, Hamilton, Ontario, Canada

[2] Department of Mathematics, Statistics and Computer Science, Faculty of Science,
St. Francis Xavier University, Antigonish, Nova Scotia, Canada

Technical Report CAS-19-09-RK
Department of Computing and Software
McMaster University

September 23, 2019

Abstract

In this paper, we introduce a framework for data-based ontology design, called Domain Information System (DIS). A DIS contains both the representation of a domain, and a data view. It offers a low coupling of these two components, through an operator that takes a datum and gives its corresponding concept. The design of the ontology component of a DIS is guided by the considered organised dataset. A DIS uses a graph and a Boolean lattice to represent the domain knowledge, and a cylindric algebra to present a view for organised datasets. We compare DIS to existing frameworks used to deal with domain and data evolution, and we show how information evolution has minimal effects on the framework.

Keywords: Formal methods, Domain knowledge representation, Ontology, Mereological ontology, Cylindric algebra, Information evolution, Data-Centric ontology design

Contents

1 Introduction

One of the main challenges for organizations is the task of transforming the huge amount of data they have stored into useful knowledge [25]. The transformation process is known as *knowledge engineering*. In order to generate knowledge from data, organizations need to interpret the data within a domain of application. A datum gets meaning once it is interpreted as a value of a certain attribute within a domain. For example, "Jan 5, 2005" is a string, and it can be seen either as the date of birth of a patient, or as the start date of employment. In the last two decades, the field of knowledge engineering saw a rise in popularity, moving from data modeling and analysis to domain knowledge representation [10]. An ontology is seen as a formal conceptualization of a domain [14], and it consists of relatively generic concepts and relations (i.e., the domain knowledge) that can be shared and reused by different applications [33].

In recent years, the field of ontologies seems to have focused on the use of Description Logic (DL) to represent a domain knowledge [24]. A DL consists of a set of concepts, individuals, and relations on them [1]. The set of concepts and roles (i.e., binary relations between concepts) are called the Terminological Box (T-Box). The set of assertions about individuals, such as mereological relations between a concept and the objects it abstracts, or relations between individuals, is called the Assertional Box (A-Box). The set of role properties and operations on roles, such as inclusion, equivalence, or composition of roles, is called the Rule Box (R-Box) [23]. As the A-Box size grows, the need to separate the T-Box and the R-Box from the A-Box becomes more and more important if systems are to be scalable [31].

In the literature, we find solutions to the challenge of decoupling the data from the conceptualization of the domain, such as Ontology Based Data Access (OBDA) systems [16, 32, 41] and DOGMA [18]. However, these approaches start from the premises that both the data source and the ontology already exist, and that they have been built independently of each other. Therefore, both the OBDA and DOGMA-based systems need to provide a mapping between the two parts. In doing so, they introduce another challenge, in the form of inefficient matching of the model of the application domain to the model of the data [41].

Building ontologies has generally been an ad-hoc process, as it does not follow a clear process. Recent years have seen developments of methodological frameworks for ontology engineering, such as Methontology [11], On-to-knowledge [35], DOGMA [18], and others [13, 19, 34, 37]. These frameworks take an agile development approach, in which engineering activities are repeated and refined over time. The conceptualisation of the domain of application is performed by domain experts with the aid of tools; however it is not an automated process. In [39], the authors mention that "despite the fact that big data research has gained rapid growth in recent years, there is a lack of grounded theories and acceptable conceptual frameworks around big data theme that enable researchers and organizations to capture the value of big data in a systematic manner." In addition, the engineering process to represent domain knowledge needs to be controlled, otherwise the resulting specification might become inconsistent with the real world knowledge [29].

Within DL, there are numerous fragments, and associated reasoning engines, which are usually optimised towards one of the three components. Some are optimised for reasoning on large T-Boxes, others for reasoning on large A-Boxes, yet others for reasoning on large R-Boxes. This fragmentation of the field of both logics and their corresponding languages brings another challenge. As an ontology evolves beyond its original scope, its representation, and implicitly the reasoning engine related to it, need to be changed as well. Consequently, there is need a for a domain representation formalism that separates data from the domain knowledge, offers support for integrating data from heterogeneous sources, and provides a more standard reasoning engine through a language close to natural

language [27].

We advance that the right approach to fill this gap is by providing a systematic, rigorous process for knowledge engineering, and to employ the use of a modular structure for domain knowledge representation. To avoid the challenges of mapping the data to existing ontologies, we propose a bottom-up knowledge engineering framework that considers the data from the onset, and uses it to guide the design of the domain representation. In this paper, we introduce a framework for data-based ontology design, called the DIS [26]. The DIS formalism consists of three parts: a domain ontology, a domain data view, and an operator that maps the two. The ontology consists of three components: a Boolean lattice, corresponding to the schema of the data, a family of rooted graphs, and a monoid of concepts. From the schema of the data, we build the core structure of the ontology, which is a Boolean lattice. Other concepts and relations pertinent to the domain are added to enrich the ontology, through the family of rooted graphs. The set of the non-lattice concepts is guided by the existence of other datasets; therefore the entire ontology is guided by the data to be considered. With this approach there is no need for expensive translations from the dataset to the ontology and vice-versa, as the atomic concepts within the ontology correspond one-to-one to attributes in the dataset. In addition, the use of cylindric algebra as a model for the data assumes an open-world approach, in which a missing data may take any possible value of the attribute it represents.

The paper is organised as follows. In Section 2, we present basic elements of a first order many-sorted language. In Section 3, we elaborate on the rationale for the DIS system, present the syntax and semantics of its language, and offer an example to discuss the process of designing a DIS-based system. In Section 5, we discuss existing challenges in the field of knowledge engineering, and show how the proposed system solves some of them. Finally, in Section 6, we point to areas of future work.

2 (First Order) Many-sorted Language

The following definitions are taken from [36]. Let S be a non-empty enumerable set of sorts. A *signature* Σ is given by the tuple $\Sigma = \langle \mathcal{F}, \mathcal{R}, r_{\mathcal{F}}, r_{\mathcal{R}} \rangle$, where $\mathcal{F}$ is a set of function symbols, $\mathcal{R}$ a set of relation symbols, $r_{\mathcal{F}}$ a mapping $r_{\mathcal{F}} : \Sigma \to S^* \times S$, assigning the rank (u,s) to each $f \in \mathcal{F}$, and $r_{\mathcal{R}}$ a mapping $r_{\mathcal{R}} : \Sigma \to S^+$, assigning the arrity u to each $R \in \mathcal{R}$. An S-ranked alphabet over Σ is given by the set of function symbols and relation symbols.

Let $u = s_1 \dots s_n$, with $s_i \in S$. A function symbol f of rank (u,s) is interpreted as an operation taking n arguments, where the i-th argument is of sort s_i, and yielding a result of sort s. A function symbol of rank (e,s), where e is the empty string, is called a *constant of sort s*. A symbol R of arrity u is interpreted as a set of n-tuples (i.e., a relation), where the i-th element of the tuple is of sort s_i.

In a many-sorted language, besides the symbols of the alphabet we can use the following symbols: (i) parenthesis, (ii) operation symbols, such as the composition of relations, (iii) logical symbols $\wedge, \vee, \implies, \forall, True,$ and $False$, and (iv) variables.

For each sort $s \in S$, we consider a countable set $\mathcal{X}_s$ of variables of sort s. Let $\mathcal{X} = \{\mathcal{X}_s\}_{s \in S}$ be the S-indexed set of variables. Terms and formulae are defined inductively. A term is (i) any constant, (ii) any variable $v \in \mathcal{X}$, or (iii) $f(t_1, \dots, t_n)$, where f is a function symbol of rank $(u,s), u = s_1 \dots s_n$, and for all $1 \leq i \leq n$, t_i is a term of sort s_i.

A (well-formed) formula is defined as follows: (i) the logical constants $True, False$ are formulae; (ii) for each relation symbol R of arrity u, where $u = s_1 \dots s_n$, and for all $1 \leq i \leq n$, if t_i is a term of sort s_i, then $R(t_1, \dots, t_n)$ is a formula; (iii) if ϕ, ψ are formulae, then $\phi \wedge \psi, \phi \vee \psi$, and $\phi \implies \psi$ are formulae; and (iv) if ϕ is a formulae, then $\forall (x \mid x \in \mathcal{X}_s : \phi(x))$[1].

[1] Throughout this paper, we adopt the uniform linear notation provided by Gries and Schneider in [12]. The general form

Given a Σ signature, a *language (of type* Σ) consists of the alphabet and a set of well formed formulae defined recursively as above.

Given an S-ranked alphabet Σ, a many-sorted Σ-structure $\mathcal{S}$ is the pair $\langle D, ^I \rangle$, where $D = \{D_s\}_{s \in S}$ is an S-indexed family of non-empty sets, and ^{I} is an interpretation function assigning functions and relations to the functions symbols and relation symbols, respectively. D_s is called the carrier set of sort s. The interpretation is as follows, for arrity $u = s_1 \dots s_n$: (i) each function symbol f of rank (u, s) is interpreted as a function $f^I : D_{s_1} \times \cdots \times D_{s_n} \to D_s$; (ii) each constant k of sort s is interpreted as an element $k^I \in D_s$; and (iii) each relation symbol R of arrity u is interpreted as a relation $R^I \subseteq D_{s_1} \times \cdots \times D_{s_n}$.

3 Domain Information Theory

In this section, we present the mathematical theory underlying the DIS. The organised dataset and the domain of application are modelled as many-sorted mathematical constructions. One of the goals in developing a new formalism for representing organised data and domain knowledge is to separate the representation aspect from the reasoning aspect. In order to achieve this, we present a language that can be used by experts to represent the domain knowledge and its link to existing organised data. Next, we present an example of a DIS.

3.1 Domain Information System: Intuitive understanding

We are interested in generating knowledge from organised datasets and domain knowledge, thus we start the construction of the DIS from the organised data. In our work, we focus on structured, organised data, represented either as a set of tuples of the same size, such as in a databse, or of different sizes, such as in log files. Therefore, we consider the elements of data to be within a Cartesian space. We believe this approach is well suited for the current state of data, as the existing unstructured data is converted to a structured format by employing machine learning and deep learning methods [20, 30].

Usually, data is stored in different formats and is in a constant state of change. While relational databases are currently the preferred method of data storage and offer great scalability, they have some drawbacks. One is that relational datasets assume a closed-world[2], in which only explicit data is recognized [27]. Since its introduction in [6], Codd's relational model of data has been accepted as a clear and succinct model for relational databases. In [17], the authors show there exists a natural embedding of a relational algebra into a diagonal-free cylindric set algebra. Moreover, data sets with records of different lengths can be modelled by the same cylindric algebra. Thus, we use a diagonal free cylindric algebra as our data view, formally described by $\mathcal{A} = \big(A, +, \star, -, 0_A, 1_A, \{\mathbf{c}_\kappa\}_{\kappa \in \mathcal{U}}\big)$. An element $a \in A$ is a tuple (or record) of an arbitrary length. Each attribute of the dataset is given as a set of values, called a sort. The set of all sorts form the universe of A, denoted by the countable set $\mathcal{U} = \{S_1, \dots, S_n\}$. The elements of A are pieces of data, with the $\star$ operator seen as a combination

of the notation is $\star(x \mid R : P)$ where $\star$ is the quantifier, x is the dummy or quantified variable, R is predicate representing the range, and P is an expression representing the body of the quantification.

[2]Relational datasets are based on the Closed-world assumption (CWA). They can be interpreted in an Open-world assumption (OWA) manner, by enriching the existing tuples with probabilistic detail elements [5]. The dataset itself is governed by the CWA principle, thus the absence of data is interpreted as negative data. E.g., in a dataset about personal information, a record corresponding to a person "John" has no value for telephone number attribute. In a closed world, this is interpreted as "John has no telephone", while in an open world it is interpreted as "There is not enough information about John's telephone".

of data elements, similar to the join operator in relation algebra. With this understanding, the 0_A is equivalent to the absence of data.

To achieve the transformation from data to information and further to knowledge, we need to give meaning to the data, specifically to its attributes or sorts. Thus we take the sorts of the dataset and consider them in a specific context. We call this context the domain of application for the data, and within it we capture the relevant concepts. We start by considering the sorts of the dataset to be a subset of the concepts of the domain, then we ask domain experts to capture other concepts that are related to the concepts originating from the dataset. In the literature, a concept is presented as a composite entity with attributes [2, 4]. This approach works well for describing a domain, however it poses a problem when existing data need to be linked to the domain representation. The main relation within an organised dataset is given by the mereological `partOf` relation.

The link between the two components is given immediately: the atomic concepts in the core construct of the ontology, which is its Boolean lattice, correspond one-to-one to the sorts of the universe of A. The operator τ maps the data elements in A to their corresponding concept in the Boolean lattice. In the following section, we give an overview of the formalism behind a DIS.

3.2 Domain Information System: Mathematical Overview

A rooted graph at t, $G_i^t = \big(C_i, R_i, t\big)$, is a connected directed graph of concepts with a unique sink $t \in C_i$. We call t the *root* of G_i^t, and define it as follows:

$$t \in C_i \text{ is } \textit{root} \text{ of } G_i^t \iff \forall (k \mid k \in C_i : k = t \vee (k,t) \in {R_i}^+) \tag{1}$$

Definition 3.1 (Domain Information System) *Let $\mathcal{C} = \big(C, \oplus, e_c\big)$ be a commutative idempotent monoid. Let $\mathcal{L} = \big(L, \sqsubseteq_c\big)$ be a Boolean lattice, with $L \subseteq C$, such that $e_c \in L$. Let I be a finite set of indices, and $\mathcal{G} = \{G_i^t \stackrel{\text{def}}{=} \big(C_i, R_i, t\big)\}_{i \in I}$ be a set of finite rooted graphs at $t \in L \cap C_i$, $C_i \subseteq C$, and $R_i \subseteq C_i \times C_i$.*

*A **domain ontology** is a mathematical structure $O \stackrel{\text{def}}{=} \big(\mathcal{C}, \mathcal{L}, \mathcal{G}\big)$. A **domain data view** is a diagonal-free cylindric algebra $\mathcal{A} = \big(A, +, \star, -, 0_A, 1_A, \{\mathbf{c}_\kappa\}_{\kappa \in L}\big)$. Let $\tau : A \to L$ be a function that maps the elements of A to their corresponding concepts in the Boolean lattice of O.*

*A **Domain Information System (DIS)** is a structure $\mathcal{D} = \big(O, \mathcal{A}, \tau\big)$.*

The operator $\oplus$ is called the composition operator, and, when restricted to the Boolean lattice, it is the join operator. The relation $\sqsubseteq_c$ is called the part of relation that we denote by `partOf`, and it is defined as the natural order on the monoid $\mathcal{C}$: $k \sqsubseteq_c k' \stackrel{\text{def}}{\iff} k \oplus k' = k'$. A concept is called *atomic* if it has no subparts, i.e.,: k is atomic $\stackrel{\text{def}}{\iff} \forall (k' \mid k' \in L : k' \sqsubseteq_c k \implies k' = k \vee k' = e_c)$.

The set of atomic concepts in the Boolean lattice $\mathcal{L}$ is denoted by $At_{\mathcal{L}}$. In this context, $\mathcal{L}$ is the free Boolean lattice generated over the set $At_{\mathcal{L}} \cup \{e_c\}$ by the composition operator $\oplus$. The set of atomic concepts is finite, as it corresponds to the attributes in the dataset, and the composition is idempotent (up to an isomorphism), thus the Boolean lattice $\mathcal{L}$ is finite. Within the lattice, the composition operator $\oplus$ allows the formation of new concepts by composing existing concepts. When we write $k \stackrel{\text{def}}{=} k_1 \oplus k_2$ we mean that "Concept k is given by the Cartesian construction of concepts k_1, k_2". Consequently, a concept k is a `partOf` another concept k', if k is a Cartesian projection of k'. For every $\kappa, \kappa' \in L$, we denote by $\kappa \backslash \kappa'$ the combination of all atomic concepts that are both `partOf` κ and not `partOf` κ', as follows:

$$\kappa \backslash \kappa' \stackrel{\text{def}}{=} \oplus (\lambda \mid \lambda \in At_{\mathcal{L}} : \lambda \sqsubseteq_c \kappa \wedge \neg(\lambda \sqsubseteq_c \kappa'))$$

On the domain ontology, a relation R_i can be extended using the lattice structure, thus extending the ontology. From a relation R_i we can infer other relations between elements of C_i and elements of L.

Definition 3.2 (Extended relation) *Let $O = \big(\mathcal{C}, \mathcal{L}, \mathcal{G}\big)$ be a domain ontology and let $G_i^t = \big(C_i, R_i, t\big) \in \mathcal{G}$ be rooted graph, with $t \in L$ its root. We call the lattice extension of R_i, the relation $R_i{}^{\uparrow} = R_i \cup R_i'$, where $R_i' = \{(k,t') \mid k \in C_i \wedge t' \in L \wedge (k,t) \in R_i \wedge t \sqsubseteq_C t'\}$.*

On the data view, $\big(A, +, \star, -, 0_A, 1_A, \{\mathbf{c}_\kappa\}_{\kappa \in L}\big)$, we define an ordering relation $\leq$, as follows: $a \leq b \stackrel{\text{def}}{=} a + b = b$. The cylindrification operators are indexed by the elements of the carrier set L of the Boolean lattice of the domain ontology. When we cylindrify on a composite concept that is formed by the Cartesian product of atomic concepts, the cylindrification is performed on all the atomic concepts of the composite concept. Hence, the cylindrification operators are defined on any concept of the Boolean lattice, as follows:

$$\mathbf{c}_\kappa x \stackrel{\text{def}}{=} \mathbf{c}_{\kappa_1} \mathbf{c}_{\kappa_2} \ldots \mathbf{c}_{\kappa_n} x, \text{ where } \kappa \in C, \kappa_i \in At_{\mathcal{L}}, \text{ and } \kappa = \mathop{\oplus}_{1 \leq i \leq n} \kappa_i \tag{2}$$

When we cylindrify an element $a \in A$ on any index $\kappa \in L$, we might extend the structure of a with the concept κ, as exemplified in Section 3.4. Since $\tau(\mathbf{c}_\kappa(a)) = \kappa \oplus \tau(a)$, where $a \in A$ and $\kappa \in L$, then for the case where $\kappa \sqsubseteq_C \tau(a)$, we get $\tau(\mathbf{c}_\kappa(a)) = \tau(a)$.

On the elements $a \in A$ we define a *focusing* operator, to obtain parts of a tuple, similar to the projection operator in relation algebra. The focusing operator is defined using the cylindrification operator, as described below. We adopt the notation $a.\kappa$ for indicating the focusing of the tuple $a \in A$ on the dimension κ. If $\neg(\kappa \sqsubseteq_C \tau(a))$, focusing $a.\kappa$ will simply be the zero of A. For any $a \in A$ and $\kappa \in L$, the focusing of datum a on κ is defined as follows:

$$a.\kappa \stackrel{\text{def}}{=} \begin{cases} \{b \mid b \in A \,:\, \tau(b) = \kappa \wedge \mathbf{c}_{(\tau(a) \setminus \kappa)} a = \mathbf{c}_{(\tau(a) \setminus \kappa)} b\}, & \text{if } \kappa \sqsubseteq_C \tau(a) \\ 0_A, & \text{otherwise} \end{cases}$$

3.3 Domain Information System: Syntax

So far, we have only captured structural information about the domain, information that can be represented using a graph-like construction. We would like to enrich this construction with algebraic statements, such as axioms and equations, to obtain a full theory for representing domain knowledge systems. For this purpose, we present the DIS formal language, as well as the axioms of the theory of DIS.

The DIS is a three-sorted theory, with its alphabet based on the set $S = \{C, L, A\}$ of sort. The non-logical symbols of the alphabet are described below.

Definition 3.3 (DIS Signature) *Let $\mathcal{D} = \big(O, \mathcal{A}, \tau\big)$ be a Domain Information System. The signature Σ_{DIS} is given by the tuple $\Sigma_{DIS} = \langle \mathcal{F}, \mathcal{R}, r_{\mathcal{F}}, r_{\mathcal{R}} \rangle$, where $\mathcal{F} = \{\oplus, \otimes, \sim, e_C, \top_{\mathcal{L}}, +, \star, -, \tau, cyl\}$ is the set of function symbols; $\mathcal{R} = \{\sqsubseteq_C, \leq\} \cup \{R_i\}_{i \in I}$ is the set of relation symbols; and $r_{\mathcal{F}}, r_{\mathcal{R}}$ are the arrity mappings, defined as follows*

$$\begin{array}{rrr} r_{\mathcal{F}}(\oplus) = (C.C, C) & r_{\mathcal{F}}(\otimes) = (L.L, L) & r_{\mathcal{F}}(\tau) = (A, L) \\ r_{\mathcal{F}}(+) = r_{\mathcal{F}}(\star) = (A.A, A) & r_{\mathcal{F}}(cyl) = (A.L, A) & r_{\mathcal{R}}(\sqsubseteq_C) = (L, L) \\ r_{\mathcal{F}}(e_C) = r_{\mathcal{F}}(\top_{\mathcal{L}}) = (e, L) & r_{\mathcal{F}}(\sim) = (L, L) & r_{\mathcal{R}}(R_i) = (C, C) \\ r_{\mathcal{F}}(0_A) = r_{\mathcal{F}}(1_A) = (e, A) & r_{\mathcal{F}}(-) = (A, A) & r_{\mathcal{R}}(\leq) = (A, A) \end{array}$$

In addition, we define three countable sets of variables: (i) $\mathcal{X}^C$, the set of variables of sort C that we denote by k, k_1, k_2, etc.; (ii) $\mathcal{X}^L$, the set of variables of sort L that we denote by $\kappa, \lambda, \kappa_1, \kappa_2$, etc.; and (iii) $\mathcal{X}^A$, the set of variables of sort A that we denote by a, b, a_1, a_2, etc. Let $\mathcal{X} = \{\mathcal{X}_s\}_{s \in S}$ be the S-indexed set, for $S = \{C, L, A\}$. The non-logical symbols of the DIS-based language are given by the functions $f \in \mathcal{F}$, relations $R \in \mathcal{R}$, and variables in $\mathcal{X}$. In addition to the S-ranked alphabet Σ_{DIS} described above, the language contains the following symbols: (i) parenthesis and brackets: "(", ")", "[", "]"; (ii) relational symbols, such as :, the composition of relations; and (iii) logical symbols $\wedge, \vee, \neg, \implies, \forall, \exists, True,$ and $False$.

The DIS-based Ontology expressions (terms and formulae) are built over this language, and are built inductively as described in Section 2. For ease of reading, we denote $cyl(\kappa, a)$ by $\mathbf{c}_\kappa(a)$. The theory of a DIS is the S-ranked alphabet Σ described above, together with the following axioms:

Let $k, k_1, k_2 \in C, \kappa, \lambda, \in L, i \in I, G_i^t \in \mathcal{G}$, with $G_i^t = \left(C_i, R_i, t\right)$

(A1) $\left(C, \oplus, e_C\right)$ is a commutative idempotent monoid

(A2) $\left(L, \oplus, \otimes, \sim, e_C, \top_{\mathcal{L}}\right)$ is a Boolean algebra

(A3) $\kappa \in L \implies \kappa \in C$

(A4) $\kappa \sqsubseteq_C \lambda \iff \kappa \oplus \lambda = \lambda$

(A5) $k \in C_i \implies k \in C$

(A6) $(k_1, k_2) \in R_i \implies k_1, k_2 \in C_i$

(A7) $(k_1, k_2) \in R_i \implies k_1 \neq k_2)$

(A8) $t \in L \wedge \forall(k \mid k \in C_i : k = t \vee (k, t) \in {R_i}^+)$

(A9) $(k_1, k_2) \in R_i \implies k_2 = t \vee (k_2, t) \in R_i^+$

(A10) $k \in C \implies k \in L \vee \exists(i \in I \mid G_i^t = \left(C_i, R_i, t\right) \in \mathcal{G} : k \in C_i)$

(A11) $\left(A, +, 1, -, 0_A, 1_A\right)$ is a Boolean algebra

(A12) $a \leq b \iff a + b = b$

(A13) $\mathbf{c}_\kappa(0) = 0$

(A14) $a \leq \mathbf{c}_\kappa(a)$

(A15) $\mathbf{c}_\kappa(a \star \mathbf{c}_\kappa(b)) = \mathbf{c}_\kappa(a) \star \mathbf{c}_\kappa(b)$

(A16) $\mathbf{c}_\kappa(\mathbf{c}_\lambda(a)) = \mathbf{c}_\lambda(\mathbf{c}_\kappa(a))$

(A17) $\tau(0_A) = \top_{\mathcal{L}}$

(A18) $\tau(1_A) = e_C$

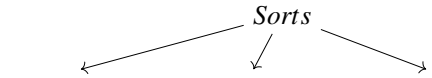
(A19) $\tau(a \star b) = \tau(a) \oplus \tau(b)$

Axiom (A1) describes the commutative idempotent monoid $\mathcal{C}$. Axioms (A2)$-$(A4) describe the Boolean lattice $\mathcal{L}$, including the partial order $\sqsubseteq_C$. Axioms (A5)$-$(A10) describe the family of rooted graphs $\mathcal{G}$, as well as the boundary we set on C. Axioms (A11)$-$(A16) describe the cylindric algebra $\mathcal{A}$. Axioms (A17)$-$(A19) define the operator τ. Using the language of DIS and the operators described in Section 3.2, domain experts can define additional concepts through axioms, as illustrated in Section 3.4.

3.4 Domain Information System: Example

For this example we use a Wine dataset, described in Figure 1. The attributes of the dataset correspond to the *atomic concepts* of the Boolean lattice $\mathcal{L}$.

Let $At_{\mathcal{L}} = \{Grape, Colour, Sugar, Body\}$ be the set of atomic concepts. Note: $e_C \in L$ and $\top_{\mathcal{L}} = Grape \oplus Colour \oplus Sugar \oplus Body$.



Grape	Colour	Sugar	Body
Merlot	Red	Dry	Medium
Pinot Grigio	White	Dry	Light
Riesling	White	Semi-sweet	Light
...	...	...	...

Figure 1: Wine dataset example

Within the Wine domain, there are a number of composite concepts available to the domain experts, as described below:

$$\begin{aligned} \mathit{Taste} &::= \mathit{Sugar} \oplus \mathit{Body} \\ \mathit{Style} &::= \mathit{Colour} \oplus \mathit{Taste} \\ \mathit{Mouthfeel} &::= \mathit{Grape} \oplus \mathit{Taste} \\ \mathit{Wine} &::= \mathit{Grape} \oplus \mathit{Colour} \oplus \mathit{Sugar} \oplus \mathit{Body} \\ \mathit{Wine} &::= \top_{\mathcal{L}} \end{aligned}$$

The elements of A are given as tuples from the dataset. For example, given the dataset in Figure 1, the element $a = \langle \mathit{Riesling}, \mathit{White} \rangle$ has $\tau(a) = \mathit{Grape} \oplus \mathit{Colour}$. The cylindrification of a on the *Body* dimension will extend the structure of a with the new dimension: $\mathbf{c}_{\mathit{Body}}(a) = \{\langle \mathit{Riesling}, \mathit{White}, \mathit{Light} \rangle, \langle \mathit{Riesling}, \mathit{White},$ and $\tau(\mathbf{c}_{\mathit{Body}}(a)) = \tau(a) \oplus \mathit{Body}$. This corresponds to the open-world assumption. If the data does not contain information on an attribute, we provide a method of replacing the missing data with possible values corresponding to that attribute. The cylindrification of a on the *Grape* dimension will not extend the structure of a. It will only extend the value of the attribute *Grape* with all the possible values found in the dataset, while preserving the values of the remainder of the attributes: $\mathbf{c}_{\mathit{Grape}}(a) = \{\langle \mathit{Merlot}, \mathit{White} \rangle, \langle \mathit{PinotGrigio}, \mathit{White} \rangle, \langle \mathit{Riesling}, \mathit{White} \rangle\}$, with $\tau(\mathbf{c}_{\mathit{Grape}}(a)) = \tau(a)$.

In Figure 2, we show a partial view of the G_1^{Grape} rooted graph, as the shaded area. The relation $\mathtt{R}_{\mathtt{grows}}$ described by the graph G_1^{Grape} is depicted with dashed lines. The graph is defined as $G_1^{\mathit{Grape}} = \langle C_1, \mathtt{R}_{\mathtt{grows}}, \mathit{Grape} \rangle$, where $C_1 = \{\mathit{Grape}, \mathit{Estate}, \mathit{Person}\}$.

A rooted graph can be further enriched by forming new relations, through the composition of existing relations. For example, consider the two relations shown in Figure 2, $\mathtt{R}_{\mathtt{grows}}$ and $\mathtt{R}_{\mathtt{contains}}$, depicted with dashed and dotted lines, respectively. The domain expert describes that the *Climate Zone* concept is linked to the *Grape* concept by defining a new relation, $\mathtt{R}_{\mathtt{possiblyGrows}}$, depicted with dotted-dashed lines. The relation $\mathtt{R}_{\mathtt{possiblyGrows}}$ is defined as follows:

$$(k_1, k_2) \in \mathtt{R}^{+}_{\mathtt{grows}} \wedge (k_2, k_3) \in \mathtt{R}_{\mathtt{contains}} \implies (k_1, k_3) \in \mathtt{R}_{\mathtt{possiblyGrows}}$$

The set of concepts in the Wine DIS is $C = L \cup C_1 \cup \{\mathit{Region}, \mathit{Climate\ Zone}\}$.

In a DIS, we do not use rooted graphs to describe the `isA` relations. Instead, they are described using specializations of the lattice concepts, such as *Red-Wine* or *Light-body White Wine*. These concepts are defined through axioms using concepts of the domain ontology and values from the data

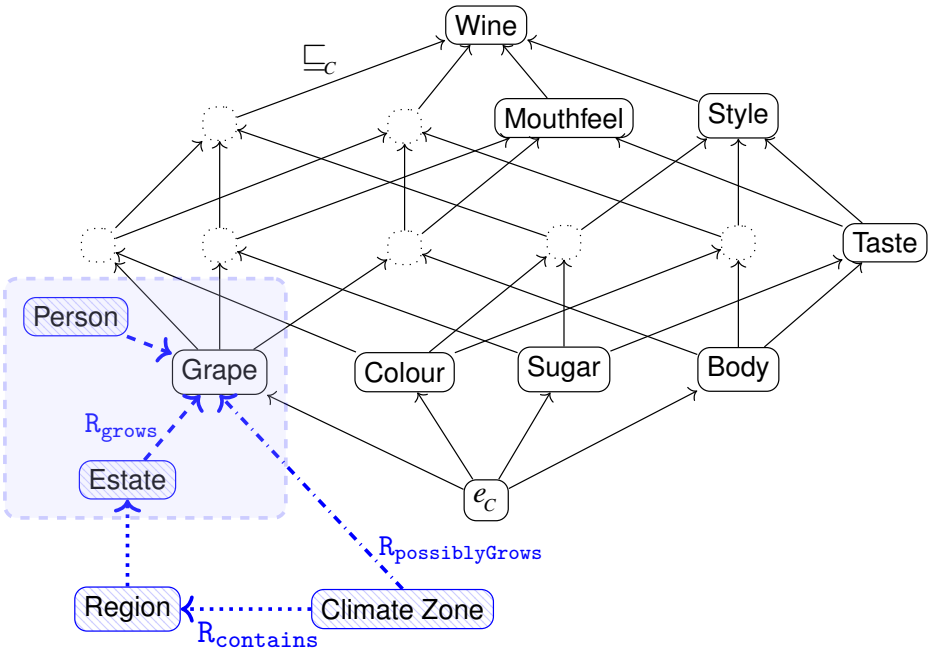


Figure 2: DIS Wine Ontology::Abstract Ontology

view, as follows:

$$\begin{aligned} \textit{Red-Wine} &= \{a \mid \tau(a) = \textit{Wine} \wedge a.\textit{Colour} = \text{`}\textit{Red}\text{'}\} \\ \textit{Light-body White Wine} &= \{a \mid \tau(a) = \textit{Wine} \wedge a.\textit{Body} = \text{`}\textit{Light}\text{'} \wedge \\ &\qquad a.\textit{Colour} = \text{`}\textit{White}\text{'}\} \end{aligned}$$

The job of the domain expert is to formally define the links to capture tacit knowledge. They can do this by using extended relations. In Figure 3, we give a reasoning example in the Wine DIS. The Boolean lattice shows a simplified view of the dataset presented in Figure 1, describing the `partOf` relation. The relation $\mathtt{R}_{\mathtt{grows}}$ links concepts in the ontology, such as *Estate*, to the root of the graph *Grape*. This relation can be extended to any concept in the lattice for which the *Grape* is a `partOf` (i.e., any concept $\kappa \in L$ s.t. $\textit{Grape} \sqsubseteq_C \kappa$). As a result, the relation $\mathtt{R}^{\uparrow}_{\mathtt{grows}}$ links *Estate* to *Wine*. By defining the concept *WineProducer* in terms of this extended relation, we can infer that *Estate* `isA` *WineProducer*, as follows:

$$(\textit{Estate}, \textit{Grape}) \in \mathtt{R}_{\mathtt{Grows}} \tag{3}$$

$$\textit{Grape} \sqsubseteq_C \textit{Wine} \tag{4}$$

$$\textit{WineProducer} = \{k \mid k \in C \wedge (k, \textit{Wine}) \in \mathtt{R}^{\uparrow}_{\mathtt{grows}}\} \tag{5}$$

From (3) and (4), a DIS-based reasoning system infers

$$(\textit{Estate}, \textit{Wine}) \in \mathtt{R}^{\uparrow}_{\mathtt{Grows}} \tag{6}$$

From (5) and (6), it is immediate that

$$\textit{Estate} \in \textit{WineProducer} \tag{7}$$

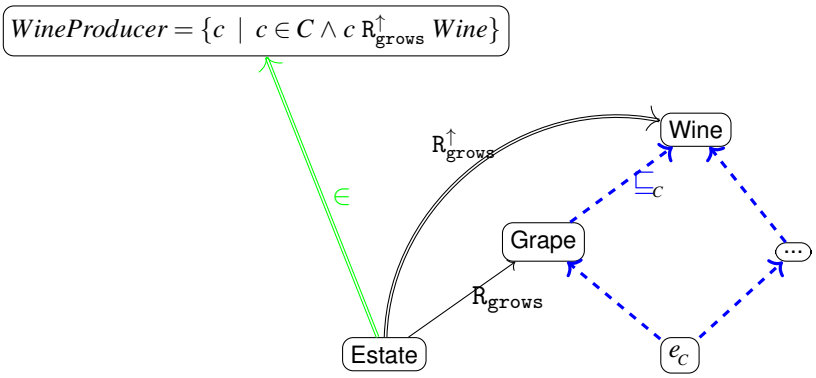


Figure 3: Wine Ontology::Wine Producer in DIS

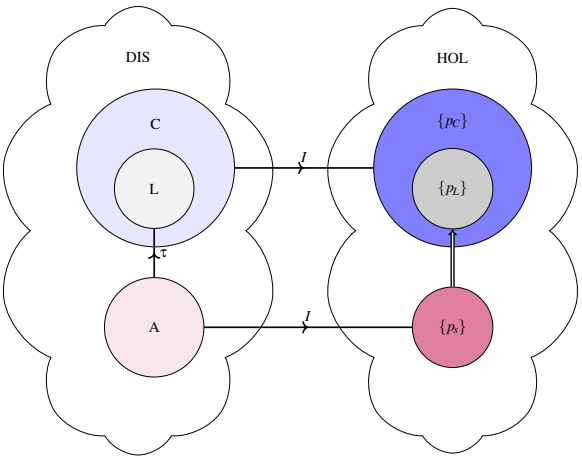


Figure 4: Relationships between elements of DIS and their interpretation

4 Domain Information System: A Model

The DIS theory described in Section 3.3 is an abstract mathematical construction. In this section, we describe a many-sorted structure based on classic logical theories and we show it is a model for the DIS.

An overview of the DIS and its proposed model is shown in Figure 4. The intuition behind the proposed interpretation is based on the fact that, in our work, the universe of discourse is given by a set of records (tuples of values) in organised datasets. The tuples correspond to concepts in the domain, and an attribute in the dataset represents a property of some concepts in the domain, interpreted as the characteristic predicate of the attribute [12]. For reasons we have discussed in Section 3.2, we take the attributes as primitive (atomic) concepts.

The separation of the data-based domain knowledge into a structural component (the Boolean lattice) and a full-valued component (the diagonal-free cylindric algebra) enables us to take two different approaches for interpretation. For the cylindric algebra we use the first order logic (FOL) interpretation presented by Tarski [15], in which the cylindrification operator ranges over a countable set of indices. An index κ is an attribute of the organised dataset, represented by a sort S_κ, or the set of values for said attribute. Each sort is thus interpreted by a predicate considered from the *extensional*

perspective. For the domain ontology we give a third order logic interpretation, in which concepts are interpreted as predicates as well. However in this case we are only interested in the *intensional* perspective of the predicates, which gives a domain-based meaning for the sorts and other concepts.

For instance, taking the example in Section 3.4, the *Grape* concept may be interpreted as *The name of a grape*. We consider this to be the intensional interpretation, given by the predicate $p_{Grape}(x) =$ "x is a grape name". A possible extensional interpretation, guided by the existing dataset, of the *Grape* concept may be given by the predicate $q_{Grape}(x) =$ "$x \in \{$Merlot, Pinot Grigio, Riesling$\}$".

In the ontology component of the DIS, we represent relations on concepts, by giving the characteristic predicate of the relation. In the model we present, the concepts are predicates, therefore relation on concepts are interpreted as predicates on predicate, thus making our model a third order logic model.

For the many-sorted Σ defined in Section 3.3, we now describe a many-sorted structure, $\mathcal{S} = \langle D, I \rangle$, where $D = \{D_C, D_L, D_A\}$, with D_C, D_L, and D_A each a set of predicates, $D_L \subseteq D_C$. I is the interpretation function that maps elements of S to elements of D, and the operators and relators of Σ to logical symbols.

In the remainder of this section we will show that third order logic, a higher order extension of FOL, is an appropriate interpretation for the DIS. Recall that the domain ontology is the mathematical structure $O = \big(\mathcal{C}, \mathcal{L}, \mathcal{G}\big)$, where $\mathcal{C} = \big(C, \oplus, e_c\big)$ is a commutative idempotent monoid, $\mathcal{L} = \big(L, \sqsubseteq_c\big)_{L \subseteq C}$ is a Boolean lattice, and $\mathcal{G} = \{G_i^t\}_{i \in I}$, a set of graphs rooted at $t \in L$.

4.1 Model for the concept structure $\mathcal{C}$

As discussed in the previous sections, the concepts in the ontology are interpreted as predicates. They are predicates that characterise the values within each of the sorts.

The interpretation of a concept $k \in C$, denoted by $I(k)$, is given by the characteristic predicate of k, $I(k) \stackrel{\text{def}}{\iff} p_k$. The set of concepts C is interpreted as a set of predicates, denoted by $I(C) = \{k \mid k \in C : I(k)\}$.

The concepts combination operator, $\oplus$, is interpreted as the logical operator conjunction, denoted by $\wedge$, as follows:

$$\forall k, k' \in C.\ I(k \oplus k') \stackrel{\text{def}}{\iff} I(k) \wedge I(k') \tag{8}$$

The set of concepts C is closed under the combination operator $\oplus$, and it is immediate that $I(C)$ is closed under $\wedge$.

$$\begin{array}{ll}
& \forall (k, k' \mid k, k' \in C : (k \oplus k') \in C) \\
\Longrightarrow & \quad \langle \text{ Applying the interpretation function } \rangle \\
& \forall (k, k' \mid k, k' \in C : I(k \oplus k') \in I(C)) \\
\Longleftrightarrow & \quad \langle \text{ Definition of the interpretation of } \oplus \text{ operator, } (8) \rangle \\
& \forall (k, k' \mid k, k' \in C : (I(k) \wedge I(k')) \in I(C)) \\
\Longrightarrow & \quad \langle\ k \in C \implies I(k) \in I(C)\ \rangle \\
& \forall (k, k' \mid I(k), I(k') \in I(C) : (I(k) \wedge I(k')) \in I(C)) \\
\Longleftrightarrow & \quad \langle \text{ Bound for renaming } I(k) = p, I(k') = q \rangle \\
& \forall (p, q \mid p, q \in I(C) : (p \wedge k') \in I(C))
\end{array}$$

The neutral element e_C is interpreted as follows:

$$I(e_C) \overset{\text{def}}{\Longleftrightarrow} True \tag{9}$$

It is immediate that $I(C) = \big(I(C), I(\oplus), I(e_C)\big) = \big(I(C), \wedge, True\big)$ is a commutative idempotent monoid. The logical operator $\wedge$ is associative, commutative, and idempotent, and $True$ is the neutral element for it.

The intuition behind this interpretation is that the more details we provide for the description of a world, the more we restrict it. For instance, in the domain of wine, the world described by the predicate p_{Colour} could include a variety of colours (including colour of wines or wine bottles). The world described by the predicate $p_{Grape} \wedge p_{Colour}$ is restricted to a descriptor of wines (represented by tuples with two values, *Grape* and *Colour*). With this interpretation, the world described by the empty concept (the pseudo-concept e_C) is completely unrestricted, there are no details provided. The predicate that characterises all the entities in the domain is the special element $True$.

4.2 Model for the Boolean lattice $\mathcal{L}$

Intuitively, if the combination operator is interpreted as the logical operator conjunction, the relation `partOf` is interpreted as follows:

$$\forall k, k' \in C.\ I(k \sqsubseteq_C k') \overset{\text{def}}{\Longleftrightarrow} (I(k') \Longrightarrow I(k)) \tag{10}$$

For convenience and readability, we define the *converse consequence* operator as $(p \Longleftarrow q) \overset{\text{def}}{\Longleftrightarrow} (q \Longrightarrow p)$.

Let L be the carrier set of the lattice $\mathcal{L}$. On the set $I(C)$, we fix the interpretation of L as the subset $I(L) \overset{\text{def}}{\Longleftrightarrow} \{k \mid k \in L : I(k)\}$. On $I(L)$, we interpret the set of atoms as $I(At_{\mathcal{L}}) = \{k \mid k \in At_{\mathcal{L}} : I(k)\}$. The top and bottom concepts are interpreted as follows:

$$I(\top_{\mathcal{L}}) \overset{\text{def}}{\Longleftrightarrow} \bigwedge (p \in I(At_{\mathcal{L}}) \mid\!: p) \tag{11}$$

$$I(\bot_{\mathcal{L}}) \overset{\text{def}}{\Longleftrightarrow} I(e_C) \tag{12}$$

It is immediate that $I(\top_{\mathcal{L}}), I(\bot_{\mathcal{L}}) \in I(L)$.

Figure 5 shows the Boolean lattice built over the set of atomic concepts {*Grape*, *Sugar*, *Body*}. On the right side, we show its interpretation, where p_1, p_2, p_3 are the characteristic predicate for the concepts *Grape*, *Sugar*, *Body*, respectively.

We need to show that the structure $I(\mathcal{L}) = \big(I(L), \Longleftarrow\big)$ is a Boolean lattice. A Boolean lattice is defined using the algebraic form, therefore we first give the algebraic lattice that is "isomorphic" to $I(\mathcal{L})$. The converse consequence operator $\Longleftarrow$ is defined using the two logical operators *And* ($\wedge$) and *Or* ($\vee$), as follows [7, Pg.28]:

$$(p \Longleftarrow q) \overset{\text{def}}{\Longleftrightarrow} (p \wedge q = q) \overset{\text{def}}{\Longleftrightarrow} (p \vee q = p) \tag{13}$$

Therefore, the algebraic lattice $\mathcal{L}' = \big(I(L), \wedge, \vee\big)$ is "isomorphic" to the relational lattice $I(\mathcal{L}) = \big(I(L), \Longleftarrow\big)$ [7]. We can show that $\mathcal{L}'$ forms a Boolean lattice. The two logical operators are associative and commutative, and they distribute over each other, $I(\bot_{\mathcal{L}})$ is the neutral element for $\vee$, $I(\top_{\mathcal{L}})$ is the neutral element for $\wedge$. The absorption law holds and the negation unitary predicate stands for complements. With this understanding, the interpretation of the Boolean lattice $\mathcal{L}$, $I(\mathcal{L})$, is a Boolean lattice.

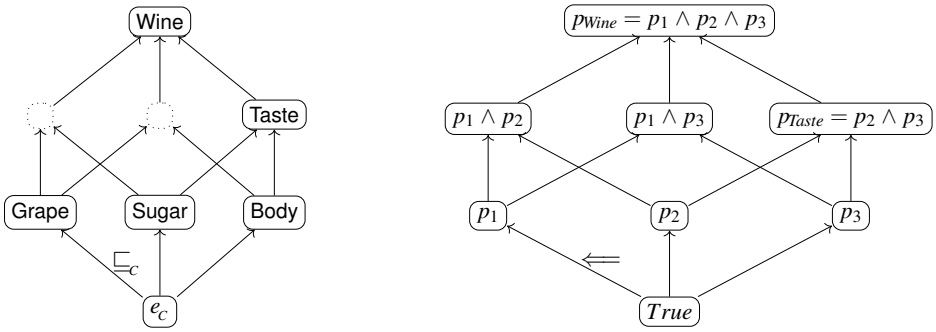


Figure 5: Interpretation of the Boolean lattice

4.3 Model for the set of rooted graphs $\mathcal{G}$

We fix a set of characteristic predicates to be disjoint from the interpretation $I(C)$, and we denote it by $I(\mathcal{R})$, where $\mathcal{R}$ is the set of relations as given in Section 3.3. The concepts are interpreted as predicates that take terms as variables, while the relations are interpreted as predicates that take other predicates as variables, therefore the two subsets are disjoint.

Given any relation $R \in \mathcal{R}$, we interpret it[3] as $I(R) \overset{\text{def}}{\Longleftrightarrow} P_R$, where $P_R \in I(\mathcal{R})$ is the characteristic predicate of the set $\{(I(c_1), I(c_2)) \mid c_1, c_2 \in C \wedge (c_1, c_2) \in R_i\}$.

The rooted graphs are based on the notion of root concept, which is defined using the transitive closure of the graph relation. We interpret the root of the graph as a unique concept that is part of the lattice and can be reached through a sequence of edges (represented by their characteristic predicate) from any other concept (vertix) in the graph.

Let $G_i^t \in \mathcal{G}, i \in I, G_i^t = \big(C_i, R_i, t\big), R_i \in \mathcal{R}, C_i = dom(R_i) \subseteq C, t \in L, p_t = I(t)$, and $P_{R_i} = I(R_i)$.

Since $G_i^t \in \mathcal{G}$, we know that $\forall(k \mid k \in C' : k = t \vee (k,t) \in {R_i}^+)$ 1. We need to show that the axiom holds for the interpretation of G_i^t. We have two distinct cases:

1. $k = t$
2. $(k,t) \in {R_i}^+$

For the case $k = t$, it is immediate that $I(k) = I(t)$ (by applying the interpretation function). For the second case, $(k,t) \in {R_i}^+$ translates into a continuous "path" from k to t, more formally $\exists(n \mid n \geq 0 : \exists(k_i \mid 1 \leq i \leq n \wedge k_i \in C_i : (k,k_1) \in R_i \wedge \cdots \wedge (k_n,t) \in R_i))$. Note that for $n = 0, (k,t) \in R_i$.

$$
\begin{aligned}
& \exists(n \mid n \geq 0 : \exists(k_i \mid 1 \leq i \leq n \wedge k_i \in C_i : (k,k_1) \in R \wedge \cdots \wedge (k_n,t) \in R_i)) \\
\Longrightarrow \quad & \langle \text{ Applying the interpretation function } \rangle \\
& \exists(n \mid n \geq 0 : \exists(I(k_i) \mid 1 \leq i \leq n \wedge I(k_i) \in I(C_i) : (I(k),I(k_1)) \in P_{R_i} \wedge \cdots \wedge (I(k_n),I(t)) \in P_{R_i})) \\
\Longleftrightarrow \quad & \langle \text{ Definition of transitive closure above } \rangle \\
& (I(k),I(t)) \in P_{R_i}^+
\end{aligned}
$$

Therefore, the interpretation of an rooted graph G_i^t is given by the construction $\big(I(C_i), I(R_i), I(t)\big) = \big(I(C_i), P_{R_i}, p_t\big)$ and it is an rooted graph itself.

[3] We adopt the usage of capital fonts for predicates that are the interpretation of relations.

The set of rooted graphs is interpreted as $I(\mathcal{G}) \stackrel{\text{def}}{\Longleftrightarrow} \{I(G_{t_i})\}_{t_i \in L}$.

With these definitions and notations, the $I(O) = \big(I(\mathcal{C}), I(\mathcal{L}), I(\mathcal{G})\big)$ is a model for the domain ontology $O = \big(\mathcal{C}, \mathcal{L}, \mathcal{G}\big)$.

4.4 Model for the domain data view $\mathcal{A}$

In [15], the authors provide a FOL interpretation for a cylindric algebra, which we adopt in our work. Recall the structure of a (diagonal-free) cylindric algebra is $\mathcal{A} = \big(A, +, \star, -, 0_A, 1_A, \{\mathbf{c}_\kappa\}_{\kappa \in L}\big)$, where the following axioms are satisfied for any $x, y \in A$, and any $\kappa, \lambda \in L$

(C1) the structure $\big(A, +, \star, -, 0_A, 1_A\big)$ is a Boolean algebra

(C2) $\mathbf{c}_\kappa 0 = 0$

(C3) $x \leq \mathbf{c}_\kappa x$

(C4) $\mathbf{c}_\kappa(x \cdot \mathbf{c}_\kappa y) = \mathbf{c}_\kappa x \cdot \mathbf{c}_\kappa y$

(C5) $\mathbf{c}_\kappa \mathbf{c}_\lambda x = \mathbf{c}_\lambda \mathbf{c}_\kappa x$

Let $\mathcal{U} = \{S_1, S_2, \dots\}$ be the finite set of disjoint sorts, where $\mid \mathcal{U} \mid = \mid At_{\mathcal{L}} \mid$, and each attribute S_i corresponds directly to an atomic concept in $At_{\mathcal{L}}$. For each sort $S_i \in \mathcal{U}$, we define its characteristic predicate p_{S_i}, as follows:

$$p_{S_i}(v) \stackrel{\text{def}}{\Longleftrightarrow} v \in S_i \tag{14}$$

The elements of A are interpreted as relations over the sorts of $\mathcal{U}$, with each relation given by a set of tuples (records) of values from the set of the attributes (or sorts) of the organised dataset. The intuition behind this interpretation is that in the concrete world, we give the characteristic predicates for the sorts strictly as their extension, i.e., the set of values for that sort found in the dataset. Unlike the more abstract world of the ontology, we are not interested that much in what is the meaning of a given attribute (sort), we only need its extension.

For simplicity, in what follows, we consider the elements of A to be sets of one tuple only. The results below can be easily extend to the general case. Let $a \in A$ be given as $a = \{\langle v_{\kappa_l}, \dots, v_{\kappa_i}, \dots, v_{\kappa_n} \rangle\}$, for some $\kappa_i \in At_{\mathcal{L}}, 1 \leq i \leq n$[4]. It is immediate that $\tau(a) = \oplus\{\kappa_i\}_{1 \leq i \leq n}$. The interpretation of $a \in A$ is denoted by its characteristic predicate $I(a) \stackrel{\text{def}}{\Longleftrightarrow} p_a$, such that

$$p_a(v_{\kappa_l}, \dots, v_{\kappa_n}) \implies \wedge\, (i \mid 1 \leq i \leq n \,:\, p_{S_{\kappa_i}}(v_{\kappa_i})) \tag{15}$$

Remark: Axiom (15) is called the "integrity constraint" in classic relational algebra.

For instance, consider the dataset depicted in Figure 1, with four attributes, *Grape*, *Colour*, *Sugar*, and *Body*. Each correspond to one sort in the domain data view, $S_{Grape} = \{Merlot, PinotGrigio, Riesling\}$, $S_{Colour} = \{Red, White\}$, etc. Example of elements $a, b \in A$ are $a = \langle Merlot, Red, Dry, Medium \rangle$ and $b = \langle Riesling, White \rangle$, and they are interpreted as the corresponding characteristic predicate, p_a, p_b respectively. Then,

$$\begin{aligned}
\tau(a) &= Grape \oplus Colour \oplus Sugar \oplus Body \\
\tau(b) &= Grape \oplus Colour \\
p_a(Merlot, Red, Dry, Medium) &\implies p_{Grape}(Merlot) \wedge p_{Colour}(Red) \wedge \\
&\qquad p_{Sugar}(Dry) \wedge p_{Body}(Medium) \\
p_b(Riesling, White) &\implies p_{Grape}(Riesling) \wedge p_{Colour}(White)
\end{aligned}$$

[4] We abuse the notation $a = \langle ... \rangle$ when it is clear from the context that a is given by a set of exactly one tuple.

The interpretation of A is denoted by $I(A) \overset{\text{def}}{\Longleftrightarrow} \breve{A}$, where $\breve{A} = \{I(a) \mid a \in A\}$.

The interpretation of the binary operator $+$ is the logical operator *or*, $\vee$, that of the binary operator $\star$ is the logical operator *and*, $\wedge$, and that of the unary operator $-$ is the logical negation, $\neg$, as follows. Let $a, b \in A$:

$$I(a+b) \overset{\text{def}}{\Longleftrightarrow} I(a) \vee I(b) \tag{16}$$

$$I(a \star b) \overset{\text{def}}{\Longleftrightarrow} I(a) \wedge I(b) \tag{17}$$

$$I(-a) \overset{\text{def}}{\Longleftrightarrow} \neg I(a) \tag{18}$$

$$I(0_A) \overset{\text{def}}{\Longleftrightarrow} \mathit{False} \tag{19}$$

$$I(1_A) \overset{\text{def}}{\Longleftrightarrow} \mathit{True} \tag{20}$$

The cylindrification operator $\mathbf{c}_\kappa$ is indexed over concepts $\kappa \in L$. For simplicity, we assume that the cylindrification operator $\mathbf{c}_\kappa a$ is applied on one dimension only, i.e., $\kappa \in At_L$. The results can be easily extended to any compound concept.There are two distinct cases:

1. Index κ is not part of the structural composition of a, i.e., $\neg(\kappa \sqsubseteq_C \tau(a))$
2. Index κ is part of the structural composition of a, i.e., $\kappa \sqsubseteq_C \tau(a)$

For instance, consider the example above, with the two elements $a, b \in A$. Cylindrification of b on dimension *Sugar* extends the structure of b and it is given by the following set of tuples:

$$\mathbf{c}_{Sugar} b = \{\langle \mathit{Riesling}, \mathit{White}, \mathit{Dry}\rangle, \langle \mathit{Riesling}, \mathit{White}, \mathit{Semi\text{-}sweet}\rangle\}$$

Cylindrification of a on dimension *Sugar* does not change the structure of the relation, it only extends its values, as follows:

$$\mathbf{c}_{Sugar} a = \{\langle \mathit{Merlot}, \mathit{Red}, \mathit{Dry}, \mathit{Medium}\rangle, \langle \mathit{Merlot}, \mathit{Red}, \mathit{Semi\text{-}sweet}, \mathit{Medium}\rangle\}$$

In the first case, the cylindrification operator executes a structural expansion, where the structure of the resulting tuples is extended with the new concept κ. The resulting data element contains tuples of the form $\langle v_{\kappa_1}, \ldots, v_{\kappa_n}, u_\kappa\rangle$, where $u_\kappa \in S_\kappa$. In the second case, the cylindrification operator executes a content expansion, where the structure of the resulting element remains unchanged. The resulting data element contains tuples of the form $\langle v_{\kappa_1}, \ldots, u_\kappa, \ldots, v_{\kappa_n}\rangle$, $\kappa = \kappa_i$ and $u_\kappa \in S_\kappa$. The value corresponding to the dimension κ_i, on which the cylindrification operator is applied to, has been replaced by the universal values of that dimension.

This intuition coincides with the open-world assumption: assuming a dimension κ for which an element $a \in A$ shows no recorded value, the cylindrification operator will extend a with all the existing values for the missing dimension κ.

The extension of $I(a)$ is the set $\{\langle v_{\kappa_1}, \ldots, v_{\kappa_n}\rangle \mid p_a(v_{\kappa_1}, \ldots, v_{\kappa_n})\}$. For the two cases above, the extension of the interpretation of the cylindrification operator $I(\mathbf{c}_\kappa a)$ is given by:

1. $\{\langle v_{\kappa_1}, \ldots, v_{\kappa_n}, u_\kappa\rangle \mid v_{\kappa_i} \in S_{\kappa_i} \wedge u_\kappa \in S_\kappa \wedge p_a(v_{\kappa_1}, \ldots, v_{\kappa_n})\}$
2. $\{\langle v_{\kappa_1}, \ldots, u_\kappa, \ldots, v_{\kappa_n}\rangle \mid v_{\kappa_i} \in S_{\kappa_i} \wedge u_\kappa \in S_\kappa \wedge p_a(v_{\kappa_1}, \ldots, v_{\kappa_n})\}$

As explained in [9], in order to obtain either of the two extensions, we add a "do not care" clause to the argument we need, in our case the dimension κ. In doing so, the cylindrification of the relation $I(a)$ along the κ index corresponds to the predicate expressed as follows:

$$I(\mathbf{c}_\kappa a) \stackrel{\text{def}}{\iff} \exists(u_\kappa \mid u_\kappa \in S_\kappa : I(a)) \tag{21}$$

It is immediate that Axiom $(C1)$ holds.

$I(\big(A,+,\star,-,0_A,1_A\big)) = \big(I(A),I(+),I(\star),I(-),I(0),I(1)\big) = \big(\check{A},\vee,\wedge,\neg,False,True\big)$ is a Boolean algebra ($\vee$ and $\wedge$ are associative, commutative, distribute over each other, the absorption law holds, $False, True$ are their respective neutral elements, and $\neg$ stands for complements).

We can show that Axiom $(C2)$ holds:

$$\begin{array}{ll}
& I(\mathbf{c}_k 0_A) \\
\iff & \quad \langle \text{ Interpretation of the cylindrification operator (21) } \rangle \\
& \exists(u_\kappa \mid u_\kappa \in S_\kappa : I(0_A)) \\
\iff & \quad \langle \text{ Interpretation of } I(0_A) \text{ (19) } \rangle \\
& \exists(u_\kappa \mid u_\kappa \in S_\kappa : False) \\
\iff & \quad \langle \exists\text{-False body } \rangle \\
& False \\
\iff & \quad \langle \text{ Interpretation of } 0_A \text{ (19) } \rangle \\
& I(0_A)
\end{array}$$

For Axiom $(C3)$ we need the interpretation of the order relation $\leq$ on tuples of data. We know that $a \leq b \iff a+b=b$, and we show that

$$I(a \leq b) \stackrel{\text{def}}{\iff} I(a) \implies I(b) \tag{22}$$

$$\begin{array}{ll}
& I(a \leq b) \\
\iff & \quad \langle \text{ Algebraic interpretation of } \leq \rangle \\
& I(a+b) = I(b) \\
\iff & \quad \langle \text{ Interpretation of operator } + \text{ (16) } \rangle \\
& I(a) \vee I(b) = I(b) \\
\iff & \quad \langle \text{ Definition of implication operator (13) } \rangle \\
& I(a) \implies I(b)
\end{array}$$

We can show that Axiom $(C3)$ holds. Let $x \in A, \kappa \in \mathcal{U}$.

$$\begin{array}{ll}
& I(x \leq \mathbf{c}_\kappa x) \\
\iff & \quad \langle \text{ Interpretation of } \leq \text{ (22) } \rangle \\
& I(x) \implies I(\mathbf{c}_\kappa x) \\
\iff & \quad \langle \text{ Interpretation of cylindrification operator (21) } \rangle \\
& I(x) \implies \exists(u_\kappa \mid u_\kappa \in S_\kappa : I(x)) \\
\iff & \quad \langle \exists\text{-Introduction axiom } (P \implies \exists x P) \rangle \\
& True
\end{array}$$

For Axiom $(C4)$, we take $x, y \in A, \kappa \in \mathcal{U}$.

$$
\begin{aligned}
& I(\mathbf{c}_\kappa(x \star \mathbf{c}_\kappa y)) \\
\iff & \quad \langle \text{ Interpretation of } \mathbf{c}_\kappa x \text{ operator (21) } \rangle \\
& \exists(u_\kappa \mid u_\kappa \in S_\kappa : I(x \star \mathbf{c}_\kappa y)) \\
\iff & \quad \langle \text{ Interpretation of } \star \text{ operator (17) } \rangle \\
& \exists(u_\kappa \mid u_\kappa \in S_\kappa : I(x) \wedge I(\mathbf{c}_\kappa y)) \\
\iff & \quad \langle \text{ Interpretation of } \mathbf{c}_\kappa y \text{ operator (21) } \rangle \\
& \exists(u_\kappa \mid u_\kappa \in S_\kappa : I(x) \wedge \exists(v_k \mid v_k \in S_\kappa : I(y))) \\
\iff & \quad \langle \text{ Distributivity of } \wedge \text{ over } \exists \ (v_k \text{ is free in } I(x)) \rangle \\
& \exists(v_k \mid v_k \in \kappa : I(y)) \wedge \exists(u_\kappa \mid u_\kappa \in S_\kappa : I(x)) \\
\iff & \quad \langle \text{ Commutativity of } \wedge \rangle \\
& \exists(u_\kappa \mid u_\kappa \in S_\kappa : I(x)) \wedge \exists(v_k \mid v_k \in \kappa : I(y)) \\
\iff & \quad \langle \text{ Interpretation of } \mathbf{c}_\kappa x \text{ operator (21) } \rangle \\
& I(\mathbf{c}_\kappa x) \wedge I(\mathbf{c}_\kappa y) \\
\iff & \quad \langle \text{ Interpretation of } \star \text{ operator (17) } \rangle \\
& I(\mathbf{c}_\kappa x \star \mathbf{c}_\kappa y)
\end{aligned}
$$

Similarly, for Axiom $(C5)$, we take $x \in A$, and $\kappa, \lambda \in \mathcal{U}$.

$$
\begin{aligned}
& I(\mathbf{c}_\kappa \mathbf{c}_\lambda x) \\
\iff & \quad \langle \text{ Interpretation of cylindrification operator, } \mathbf{c}_\kappa \text{ (21) } \rangle \\
& \exists(u_\kappa \mid u_\kappa \in S_\kappa : I(\mathbf{c}_\lambda x)) \\
\iff & \quad \langle \text{ Interpretation of cylindrification operator, } \mathbf{c}_\lambda \text{ (21) } \rangle \\
& \exists(u_\kappa \mid u_\kappa \in S_\kappa : \exists(u_\lambda \mid u_\lambda \in S_\lambda : I(x))) \\
\iff & \quad \langle \text{ Idempotency of } \wedge \rangle \\
& \exists(u_\kappa \mid u_\kappa \in S_\kappa : \exists(u_\lambda \mid u_\lambda \in S_\lambda : I(x) \wedge I(x))) \\
\iff & \quad \langle \text{ Distributivity of } \wedge \text{ over } \exists \ (u_\lambda \text{ is free in } I(x)) \rangle \\
& \exists(u_\kappa \mid u_\kappa \in S_\kappa : I(x) \wedge \exists(u_\lambda \mid u_\lambda \in S_\lambda : I(x))) \\
\iff & \quad \langle \text{ Distributivity of } \wedge \text{ over } \exists \ (u_\kappa \text{ is free in } I(x)) \rangle \\
& \exists(u_\lambda \mid u_\lambda \in S_\lambda : I(x)) \wedge \exists(u_\kappa \mid u_\kappa \in S_\kappa : I(x)) \\
\iff & \quad \langle \text{ Commutativity of } \wedge \rangle \\
& \exists(u_\kappa \mid u_\kappa \in S_\kappa : I(x)) \wedge \exists(u_\lambda \mid u_\lambda \in S_\lambda : I(x)) \\
\iff & \quad \langle \text{ Distributivity of } \wedge \text{ over } \exists \ (u_\kappa \text{ is free in } I(x)) \rangle \\
& \exists(u_\lambda \mid u_\lambda \in S_\lambda : I(x) \wedge \exists(u_\kappa \mid u_\kappa \in S_\kappa : I(x))) \\
\iff & \quad \langle \text{ Distributivity of } \wedge \text{ over } \exists \ (u_\kappa \text{ is free in } I(x)) \rangle \\
& \exists(u_\lambda \mid u_\lambda \in S_\lambda : \exists(u_\kappa \mid u_\kappa \in S_\kappa : I(x) \wedge I(x))) \\
\iff & \quad \langle \text{ Idempotency of } \wedge \rangle \\
& \exists(u_\lambda \mid u_\lambda \in S_\lambda : \exists(u_\kappa \mid u_\kappa \in S_\kappa : I(x))) \\
\iff & \quad \langle \text{ Interpretation of cylindrification operator, } \mathbf{c}_\kappa \text{ (21) } \rangle \\
& \exists(u_\lambda \mid u_\lambda \in D_\lambda : I(\mathbf{c}_\kappa x))
\end{aligned}
$$

$\iff$ $\langle$ Interpretation of cylindrification operator, $\mathbf{c}_\lambda$ (21) $\rangle$
$I(\mathbf{c}_\lambda \mathbf{c}_\kappa x)$

Therefore, Axioms $(C4)$ and $(C5)$ both hold.

With this interpretation, the mathematical structure $I(\mathcal{A}) = \big(I(A), \vee, \wedge, \neg, False, True, I(\mathbf{c}_\kappa)\big)$ is a diagonal-free cylindric algebra.

With this understanding, all five axioms of a diagonal free cylindric algebra hold:

1. $I(\big(A, +, \star, -, 0_A, 1_A\big)) = \big(I(A), \vee, \wedge, \neg, False, True\big)$ is a Boolean algebra
2. $I(\mathbf{c}_\kappa 0_A) = I(0_A)$
3. $I(x \leq \mathbf{c}_\kappa x) = True$
4. $I(\mathbf{c}_\kappa(x \star \mathbf{c}_\kappa y)) = I(\mathbf{c}_\kappa x \star \mathbf{c}_\kappa y)$
5. $I(\mathbf{c}_\kappa \mathbf{c}_\lambda x) = I(\mathbf{c}_\lambda \mathbf{c}_\kappa x)$

Therefore, the mathematical structure $I(\mathcal{A}) = \big(I(A), \vee, \wedge, \neg, False, True, I(\mathbf{c}_\kappa)\big)$ is a diagonal-free cylindric algebra.

5 Discussion

In this paper, we propose a formal process for representing the organised datasets, their domain of application, and the relation between these two aspects.

In [8], the authors recognize that in the process of analysing data, a large amount of time is spent on mapping the data to the semantic context. Considering the new characteristics of data (especially volume, variety, and velocity), the current practice of building ontologies from scratch, with no regard to the acquired and stored data, is no longer an acceptable approach. We need methodologies to support the design of domain knowledge, while taking in account the existing organised data. With the rise of the Semantic Web, work to extract knowledge from free text has been started. To the best of our knowledge there has been little done to capture the domain knowledge starting from existing structured data.

Existing knowledge representation formalisms [1, 38, 40] have one common property: if data is considered, it is included within the representation of the ontology. Thus, as the data volume increases, the ontology may become unmanageable. There is a clear need to separate the data from the domain knowledge, with well-defined links between the two components. We have proposed a modular structure to represent the domain knowledge: the domain ontology captures the conceptualisation of the considered domain, and the domain data view represent the data. The two components are loosely coupled through the τ operator. The modularity of the DIS-based systems allows for various interpretations of the same dataset in different contexts.

As discussed in Section 1, there already exists knowledge representation frameworks that decouple the data from the domain of application. OBDA systems [16, 32] are built primarily to quickly adapt to changes in the data, as well as to allow a seamless integration of different data source formats. However, OBDA systems start from the premise that both the data source and an ontology already exist, and they provide a mapping between the two parts. In doing so, the main challenge of the OBDA systems is matching the model of the application domain (i.e., the ontology) to the model of the data (i.e., its schema). In addition, the translation of user queries from the language of the ontology to the language of the data source(s) poses efficiency challenges [41]. While the literature abounds

with research on optimisation of the translation phase [3, 16, 22, 28, 32], no widely accepted solution is currently available.

In DOGMA [18], a methodological framework for ontology engineering, an ontology is doubly-articulated into a domain axiomatization (i.e., a set of concepts, relations, and axioms that describe the domain) and an application axiomatization. The double-articulation ensures that the typically local application axiomatization uses only concepts and relations already defined in the typically global domain axiomatization. This approach implies that the ontology (i.e., domain axiomatization) is built independently from the data model (i.e., application axiomatization). Thus, the challenge of the mismatch between the model of the domain and the model of the data is still present in DOGMA-based systems.

In our work, the DIS-based systems build the domain ontology from an existing dataset. This structure enables the automated extraction of the ontology core as a Boolean lattice, built over the `partOf` relation. The dataset attributes become atomic concepts, and other concepts and relations on them are captured through the set of rooted graphs in the DIS, with the assistance of domain experts. The mapping from the data view to the domain ontology is provided by process of designing the ontology, through the τ operator.

Intuitively, as new knowledge is generated from large volumes of data, it will produce large volumes of knowledge. Current formal information systems, such as Information Algebra [21], Concept Algebra [40], and E^CII [38] , offer a method for focusing or sharpening of information, which works well with the closed-world assumption of databases. In the open-world we assume in the proposed DIS, we provide a method for "expanding" the information, through the cylindrification operator. In addition, through the focusing operator described in Section 3.2, a DIS-based system offers a focusing method similar to that given in [21, 38, 40].

We advance that DIS-based systems offer a structure in which information evolution is mostly limited to the component where the change occurs. In a DIS-based system, information evolution can be initiated from three sources (1) domain, (2) data content, and (3) data structure. In the first case, domain changes are captured at the ontology level, more precisely in the rooted graphs or the axioms of the system. The data component remains consistent with the ontology, and it needs no updates, as it is not affected by a domain change. In the second case, as the data structure is not affected, the ontology structure remains unchanged, and therefore consistent with the changed data. In the third case, any changes made to the attributes of the data view will initiate a change to the Boolean lattice. The atomic concepts corresponding to the modified attributes need to be modified. However, the only consistency check that needs to be performed is on the rooted graphs and axioms using the modified Boolean lattice concepts. For example, if the data schema removes an attribute, its corresponding atomic concept in the Boolean lattice must be be removed as well. The domain expert must update the rooted graphs for which the root is the removed atomic concept or its super-parts, and the axioms that involve the removed concept. Adding attributes in the data schema will not introduce inconsistencies in the existing system. Therefore, in a DIS-based system, evolution of one component has no or minimal impact on the other components of the system.

6 Conclusion and Future Work

In this paper, we introduce a framework for data-based ontology design, called DIS. The DIS is a modular structure that makes effective use of classic mathematical structures to represent a domain of application and the data view it is based on. The only coupling between the two components is given through the τ operator. Due to the low coupling of the DIS components, information evolution

has minimal impact on a DIS. We present a language that captures both the domain data view and the domain representation in a unified manner. In addition, we describe the DIS theory and we show it accepts a model.

The immediate next step is to build a user-friendly language to specify DIS-based systems. This language needs to be close to natural language and hide all the mathematical details. Once this step is completed, we will build a tool that automatically generates the Boolean lattice component of the domain ontology from the schema of the existing organised dataset.

Our proposed structure handles one dataset only, linking it to the Boolean lattice. Another research direction that stems from this approach is the ability to handle multiple datasets, effectively zooming into the concepts in the domain ontology, by providing them structure. This could be achieved in two ways. The first approach is to extend the DIS to handle a set of Boolean lattices. The rest of the elements of the DIS, including the cylindric algebra that models the data already allows for multiple datasets. The second approach is to change the construction of the domain ontology, by building a massive Boolean lattice from all the attributes from all the datasets. Each approach should be explored to determine what extension should be chosen.

By extending the proposed information system to enable multiple Boolean lattices, we provide the concepts in the rooted graphs of the domain ontology with a more granular structure. This will enable us to build an algorithm for translating existing ontologies into a DIS, by mapping concepts from the source ontology to concepts in the destination domain ontology, and mapping attributes to atomic concepts that are the foundation of multiple Boolean lattices. The other relations in the source ontology will be mapped to rooted graphs in the destination domain ontology.

References

[1] Franz Baader, Diego Calvanese, Deborah McGuinness, Peter Patel-Schneider, and Daniele Nardi. *The description logic handbook: Theory, implementation and applications*. Cambridge university press, 2003. 1, 17

[2] Angel Barrasa. Hierarchical taxonomy of leadership behavior: antecedents, structure, and influence in work groups effectiveness. research paper, Complutense University of Madrid, Madrid, 2004. 4

[3] Meghyn Bienvenu, Stanislav Kikot, Roman Kontchakov, Vladimir V Podolskii, and Michael Zakharyaschev. Ontology-mediated queries: Combined complexity and succinctness of rewritings via circuit complexity. *Journal of the ACM (JACM)*, 65(5):28, 2018. 18

[4] Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Daniele Nardi. Reasoning in expressive description logics. *Handbook of automated reasoning*, 2:1581–1634, 2001. 4

[5] Ismail Ilkan Ceylan, Adnan Darwiche, and Guy Van den Broeck. Open-world probabilistic databases. In *Fifteenth International Conference on the Principles of Knowledge Representation and Reasoning*, 2016. 3

[6] Edgar F Codd. A relational model of data for large shared data banks. *Communications of the ACM*, 13(6):377–387, 1970. 3

[7] B.A. Davey and H.A. Priestly. *Introduction to Lattices and Order*. Cambridge University Press, 1990. 11

[8] Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, Antonella Poggi, and Riccardo Rosati. Using ontologies for semantic data integration. In *A Comprehensive Guide Through the Italian Database Research Over the Last 25 Years*, pages 187–202. Springer International Publishing, 2018. 17

[9] Anita Burdman Feferman and Solomon Feferman. *Alfred Tarski: life and logic*. Cambridge University Press, 2004. 15

[10] Dieter Fensel. Ontologies. In *Ontologies*, pages 11–18. Springer, 2001. 1

[11] Mariano Fernández-López, Asunción Gómez-Pérez, and Natalia Juristo. Methontology: from ontological art towards ontological engineering. In *Proc. Symposium on Ontological Engineering of AAAI*, 1997. 1

[12] David Gries and Fred B Schneider. *A logical approach to discrete math*. Springer Science & Business Media, 2013. 2, 9

[13] Michael Grüninger and Mark S Fox. Methodology for the design and evaluation of ontologies. In *Proceedings of the Workshop on Basic Ontological Issues in Knowledge Sharing, IJCAI-95*, 1995. 1

[14] Nicola Guarino, Daniel Oberle, and Steffen Staab. What is an ontology? In *Handbook on ontologies*, pages 1–17. Springer, 2009. 1

[15] Leon Henkin, J Donald Monk, and Alfred Tarski. Cylindric algebras–part ii, volume 115 of studies in logic and the foundations of mathematics, 1985. 9, 13

[16] Dag Hovland, Davide Lanti, Martin Rezk, and Guohui Xiao. Obda constraints for effective query answering. In *International Symposium on Rules and Rule Markup Languages for the Semantic Web*, pages 269–286. Springer, 2016. 1, 17, 18

[17] Tomasz Imieliński and Witold Lipski Jr. The relational model of data and cylindric algebras. *Journal of Computer and System Sciences*, 28(1):80–102, 1984. 3

[18] Mustafa Jarrar and Robert Meersman. Ontology engineering–the dogma approach. In *Advances in Web Semantics I*, pages 7–34. Springer, 2008. 1, 18

[19] Jason Jaskolka, Wendy MacCaull, and Ridha Khedri. Towards an ontology design architecture. In *2015 International Conference on Computational Science and Computational Intelligence (CSCI)*, pages 132–135. IEEE, 2015. 1

[20] Craig A Knoblock, Kristina Lerman, Steven Minton, and Ion Muslea. Accurately and reliably extracting data from the web: A machine learning approach. In *Intelligent exploration of the web*, pages 275–287. Springer, 2003. 3

[21] Jürg Kohlas and Robert F Stärk. Information algebras and consequence operators. *Logica Universalis*, 1(1):139–165, 2007. 18

[22] Roman Kontchakov, Martin Rezk, Mariano Rodriguez-Muro, Guohui Xiao, and Michael Zakharyaschev. Answering sparql queries over databases under owl 2 ql entailment regime. In *International Semantic Web Conference*, pages 552–567. Springer, 2014. 18

[23] Markus Krötzsch, Frantisek Simancik, and Ian Horrocks. A description logic primer. *arXiv preprint arXiv:1201.4089*, 2012. 1

[24] Andrew LeClair, Alicia Marinache, Haya Ghalayini, Ridha Khedri, and Wendy MacCaull. A systematic literature review and discussion on ontology modularization techniques. 2019. (submitted to IEEE Transactions on Knowledge and Data Engineering). 1

[25] Maurizio Lenzerini. Ontology-based data management. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, pages 5–6, 2011. 1

[26] Alicia Marinache. On the structural link between ontologies and organised data sets. Master's thesis, McMaster University, Hamilton, Ontario, 2016. 2

[27] Dorin Moldovan, Marcel Antal, Dan Valea, Claudia Pop, Tudor Cioara, Ionut Anghel, and Ioan Salomie. Tools for mapping ontologies to relational databases: A comparative evaluation. In *2015 IEEE International Conference on Intelligent Computer Communication and Processing (ICCP)*, pages 77–83. IEEE, 2015. 2, 3

[28] Jose Mora and Óscar Corcho. Engineering optimisations in query rewriting for obda. In *Proceedings of the 9th International Conference on Semantic Systems*, pages 41–48. ACM, 2013. 18

[29] Kamran Munir and M Sheraz Anjum. The use of ontologies for effective knowledge modelling and information retrieval. *Applied Computing and Informatics*, 2017. 1

[30] Maryam M Najafabadi, Flavio Villanustre, Taghi M Khoshgoftaar, Naeem Seliya, Randall Wald, and Edin Muharemagic. Deep learning applications and challenges in big data analytics. *Journal of Big Data*, 2(1):1, 2015. 3

[31] Rinton Press. Ontology and database mapping: a survey of current implementations and future directions. *Journal of Web Engineering*, 7(1):001–024, 2008. 1

[32] Juan F Sequeda, Marcelo Arenas, and Daniel P Miranker. Obda: query rewriting or materialization? in practice, both! In *International Semantic Web Conference*, pages 535–551. Springer, 2014. 1, 17, 18

[33] Peter Spyns, Robert Meersman, and Mustafa Jarrar. Data modelling versus ontology engineering. *ACM SIGMod Record*, 31(4):12–17, 2002. 1

[34] Steffen Staab, Rudi Studer, H-P Schnurr, and York Sure. Knowledge processes and ontologies. *IEEE Intelligent systems*, 16(1):26–34, 2001. 1

[35] York Sure, Steffen Staab, and Rudi Studer. On-to-knowledge methodology (otkm). In *Handbook on ontologies*, pages 117–132. Springer, 2004. 1

[36] Andrzej Tarlecki et al. Some nuances of many-sorted universal algebra: A review. *Bulletin of EATCS*, 2(104), 2013. 2

[37] Michael Uschold and Martin King. *Towards a methodology for building ontologies*. Artificial Intelligence Applications Institute, University of Edinburgh Edinburgh, 1995. 1

[38] Lidong Wang, Xiaodong Liu, and Jiannong Cao. A new algebraic structure for formal concept analysis. *Information Sciences*, 180(24):4865–4876, 2010. 17, 18

[39] Xiaojun Wang, Leroy White, and Xu Chen. Big data research for the knowledge economy: past, present, and future. *Industrial Management & Data Systems*, 115(9), 2015. 1

[40] Yingxu Wang. On concept algebra: A denotational mathematical structure for knowledge and software modeling. *International Journal of Cognitive Informatics and Natural Intelligence (IJCINI)*, 2(2):1–19, 2008. 17, 18

[41] Guohui Xiao, Diego Calvanese, Roman Kontchakov, Domenico Lembo, Antonella Poggi, Riccardo Rosati, and Michael Zakharyaschev. Ontology-based data access: a survey. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pages 5511–5519. AAAI Press, 2018. 1, 17