Toward Measuring Knowledge Loss due to Ontology Modularization

Andrew LeClair[a], Ridha Khedri and Alicia Marinache
Department of Computing and Software, McMaster University, 1280 Main Street West, Hamilton, Canada

Keywords: Ontology, Ontology Modularization, View Traversal, View Extraction, Knowledge Loss, Agent Knowledge, Agent Reasoning, Autonomous Agents.

Abstract: This paper formalizes the graphical modularization technique, View Traversal, for an ontology-based system represented using the Domain Information System (DIS). Our work is motivated by the need for autonomous agents, within an ontology-based system, to automatically create their own views of the ontology to address the problems of ontology evolution and data integration found in an enterprise setting. Through DIS, we explore specific ontologies that give Cartesian perspectives of the domain, which allows modularization to be a means for agents to extract views of specific combinations of data. The theory of ideals from Boolean algebra is used to formalize a module. Then, with the use of homomorphisms, the quantity of knowledge within the module can be measured. More specifically, through the first isomorphism theorem, we establish that the loss of information is quantified by the kernel of the homomorphism. This constitutes a foundational step towards theories related to reasoning on partial domain knowledge, and is important for applications where an agent needs to quickly extract a view that contains a specific set of knowledge.

1 INTRODUCTION

Conceptualizing ontologies and using them in an enterprise setting is a difficult task due to the problems of data integration (Ziegler and Dittrich, 2004), the co-ordination of multiple autonomous agents (Huhns and Singh, 1997), and the evolution of the domain (Benomrane et al., 2016; Dietz, 2006). Despite these problems, there does not yet exist an approach that properly addresses them. Ontology-based Data Access (OBDA) attempts to address the issues associated with data integration and ontology evolution (Poggi et al., 2008). The co-ordination of multiple agents in an ontological setting is studied in several papers (e.g., (Maedche et al., 2003; Belmonte et al., 2008; Freitas et al., 2017)).

There is a need in addressing these three tasks together such that the governing of the domain can be conducted using an ontology that uses the data in initial construction or reasoning tasks, such as in (Nadal et al., 2019). Existing paradigms, such as OBDA, are used to address the issues of evolving and heterogeneous data that come from multiple sources. However, complications arise from updates to the data. Therefore, restrictions must be placed on schema changes to not affect the ontology, or on version controls to control the domain's evolution. The problems associated with ontology evolution are further demonstrated in (Benomrane et al., 2016), which discusses the complications introduced with multiple agents and the current need of manual intervention by an ontologist.

OBDA and the other approaches formalize their ontologies using Description Logic (DL), such as in (Maedche et al., 2003; Motik, 2006; Baader, 2003; Hustadt et al., 2005; Bao et al., 2009). In (Maedche et al., 2003) several limitations to DL-based ontologies are raised. The first is the complexity of reasoning tasks increasing with the expressivity of DL fragment or the amount of data used (Motik, 2006; Baader, 2003). In (Hustadt et al., 2005), it is noted that the way the existential and universal quantifiers interact with the data can cause the time complexity of reasoning tasks to rise to exponential time. What often results is during the design phase, a less expressive fragment of DL is used to avoid the quantifiers or operators, or the amount of data the ontology utilizes is limited. Additionally, as discussed in (LeClair and Khedri, 2016), DL only allows for a single context of the concepts, which further limits the reasoning capabilities, and has resulted in the effort for being able to capture these contexts with extensions to DL such as with Package-based DL (Bao et al., 2009). However, even these are encumbered by the limited expressivity

[a] https://orcid.org/0000-0003-2779-8000

LeClair, A., Khedri, R. and Marinache, A.
Toward Measuring Knowledge Loss due to Ontology Modularization.
DOI: 10.5220/0008169301740184
In *Proceedings of the 11th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2019)*, pages 174-184
ISBN: 978-989-758-382-7

for reasoning inherent to using DL.

This work aims at addressing the issues associated with an evolving domain, rich with data, and numerous autonomous agents. To reach this goal, rather than using DL, we utilize Domain Information System (DIS) (Marinache, 2016) to formalize an ontology-based system. It is with DIS we aim to avoid monolithic ontologies (discussed in (Maedche et al., 2003)), and create an optimal ontology-based system as described in (Jaskolka et al., 2015). We aim to address domains that have a large set of constantly evolving data that must be reflected in the ontology that is being used by multiple agents for queries or other governing tasks. We also make the open world assumption, therefore, some data elements could be unspecified.

In this work we address the particular issues associated with autonomous agents interacting with an ontology-based system. DIS provides a means to formalize a procedure for the agents to extract a module from the ontology to fit their needs. The theory of ideals from Boolean algebra is used to extract smaller components that are shown to be complete modules. These customizable views are created through a process called *modularization*. In other words, the views are modules of the original DIS ontology. The underlying theory of DIS is then used to characterize an agent's potential knowledge within the system.

Ontology modularization is an active research field with the aim to ensure the ontologies are usable for tractable reasoning tasks, and adhere to established engineering principles that promote maintainability. Examples of recent modularization efforts can be seen in (Xue and Tang, 2017; Khan and Keet, 2015; Algergawy et al., 2016; Babalou et al., 2016; Movaghati and Barforoush, 2016; De Giacomo et al., 2018). These approaches vary by the ontology formalism they modularize on, the components of the ontology used to modularize (data, concepts, or both), and what types of modules they produce. The utilization of ideals to discuss modularization in the context of DL has been explored in (Del Vescovo et al., 2011), but requires rigorous computation. DIS is able to simply compute ideals using its underlying theory.

The remainder of the paper is as follows. In Section 2, we evaluate the work of utilizing ontologies in multi-agent systems for customizing views, as well as the literature regarding OBDA. In Section 3, we introduce the necessary mathematical background to facilitate discussion on the modularization process and knowledge quantification. In Section 4, we provide the findings regarding how a DIS-based ontology can be modularized. In Section 5, we discuss how said modularization can be used to characterize the potential knowledge of an agent, followed by a discussion in Section 6.

2 RELATED WORK

In (Wache et al., 2001), the authors highlight the need for the ontology of an ontology-based system to be modularizable so as to avoid the issues associated with using a monolithic ontology. In addition to the complications caused by a monolithic ontology, in (Jaskolka et al., 2015), the authors discuss the need to have the ability for several local ontologies to communicate by using a shared language. Nowadays, DL, as an ontology formalism, is the standard due to its wide usage by research teams and several implementations. However, Wache et al. (Wache et al., 2001) point to multiple limitations with the formalism such as the static nature of the ontologies created, intractability of reasoning tasks (when using expressive fragments), and tendency to become monolithic. Thus, we investigate DIS (formally introduced in Section 3) as an alternative formalism for an ontology-based system.

There exist several recent approaches to multi-agent systems that utilize an ontology at its core (e.g., (Pakdeetrakulwong et al., 2016; Kantamneni et al., 2015; Zhou et al., 2017)). These systems require agents to have their own ontology, or pieces of a shared larger ontology, that they can reason on. The design of agents having their own queries that they can answer allows for more efficient query answering as the agent only needs to refer to a smaller, more specific ontology. Additionally, it allows for the agents to collaborate, each with their respective expertise as determined by the ontology they contain, to answer more complex queries. However, for collaboration to occur, the agents must have some shared language that is provided by an additional ontology that every agent can communicate through. Although the agents can efficiently answer these small queries, any complex query that requires collaboration will thus suffer from the issues associated with monolithic ontologies.

The best way to provide each agent with the needed part of the used large ontology is through modularizing the ontology into modules that each agent will utilize. As all modules come from the same ontology, this mitigates the issue of needing another ontology to facilitate communication between agents. However, this requires an ontology that can be broken into the modules that each agent requires. In other words, this method requires an ontology that conceptualizes the entire domain (rather than a specific view for an agent). Examples of such an implementation can be found in (Anand et al., 2014; Belmonte et al., 2008). The modularization techniques that are proposed in these papers are heuristic in nature, not able to guarantee properties such as knowledge preservation or correctness. Concerns regarding the modularization process and lack of formal method arise when considering the use of ontologies that consist of hundreds or thousands of concepts, such as in (Ashburner et al., 2000; Raimond et al., 2007; Brickley and Miller, 2010; Spackman et al., 1997; Rector et al., 1996; Bodenreider, 2004; Suchanek et al., 2008).

Regardless of the design approaches used for an ontology-based system, it will require a full update (recheck for completeness, etc.) whenever an agent's ontology changes due to the domain's evolution (Benomrane et al., 2016). This can be a consuming process depending on how often the agents domain knowledge is expected to change, and can result in a design where the agents are static and seldom react to change.

In addition to the complications associated with ontology evolution, we investigate how the agents can interact with the data. OBDA is the approach used by the Semantic Web for linking data to an ontology (Poggi et al., 2008). OBDA paradigm aims to connect an ontology layer to a data layer so that rich queries can be made using the ontology, and answered using the data. However, it is not a trivial task to create an ontology from a dataset. The task is described as the bootstrapping problem (Jiménez-Ruiz et al., 2015), and OBDA is mostly considered as read-only as it puts restrictions on the ability to modify the datasets and handle updates (Xiao et al., 2018). Additionally, the query transformation process is not straightforward, depending on multiple aspects of the ontology, queries, and data consistency (Bienvenu et al., 2018).

This research seeks to provide a means for agents to automatically and systematically extract views from an ontology on-the-fly to achieve the tasks they are required to do. We seek to have only a single ontology that can be modularized to avoid a monolithic ontology existing at the higher-level. By extracting them from a single ontology, we ensure that the agents are able to communicate using the same language. We also seek to have a system that is adaptable to change and evolution, allowing for the data that the agents have to be malleable without the need for modification of the ontology itself.

3 MATHEMATICAL BACKGROUND

In this Section, we present the necessary mathematical background to communicate how the lattice and underlying Boolean algebra is utilized in DIS to conduct the modularization.

A *lattice* is an abstract structure that can be defined as either a relational or algebraic structure (Davey and Priestley, 2002). We provide each definition, as well as the connection between them, below.

If $(L, \leq)$ is a partially ordered set, we define an upper bound and lower bound as follows. For an arbitrary subset $S \subseteq L$, we define an element $u \in L$ as an upper bound of S if $s \leq u$ for each $s \in S$. Dually, we define an element $l \in L$ as a lower bound of S if $l \leq s$ for each $s \in S$. An upper bound u is defined as a least upper bound (dually, a lower bound l is defined as a greatest lower bound) if $u \leq x$ for each upper bound $x \in S$ ($x \leq l$ for each lower bound $x \in S$). A least upper bound is typically referred to as a *join*, and a greatest lower bound as a *meet*. If every two elements $a, b \in L$ have a join, then the partially ordered set is called a join-semilattice. Similarly, if every two elements $a, b \in L$ have a meet, then the partially ordered set is called a meet-semilattice. A lattice is a partially ordered set that is both a join- and meet-semilattice.

An algebraic structure $(L, \oplus, \otimes)$, which consists of a set L and the two binary operators $\oplus$ and $\otimes$ that are commutative, associative, idempotent, and satisfy the absorption law (i.e., $a \oplus (a \otimes b) = a \otimes (a \oplus b) = a$, for $a, b, c \in L$).

The relational and algebraic structure can be connected $a \leq b \iff (a = a \otimes b) \iff (b = a \oplus b)$.

The connection between the relational and algebraic definition of a lattice allows us to freely discuss relational or algebraic aspects. This is significant as we will be using both the algebraic and relational definitions interchangeably in the following sections and that for simplicity. We will also require the notion of a *sublattice*, which is simply defined as a nonempty subset M of a lattice L that satisfies $x \oplus y \in M$ and $x \otimes y \in M$ for all $x, y \in M$. In other words, a sublattice is a subset of the lattice in which all joins and meets are preserved in the subset.

We now introduce a distinguished lattice: the Boolean lattice (Sikorski et al., 1969). A Boolean lattice is defined as a *complemented distributive* lattice. A complemented lattice is one that is bounded (includes a *top* concept ($\top$) and a *bottom* concept ($\bot$)), and every element a has a *complement* (an element b satisfying $a \oplus b = \top$ and $a \otimes b = \bot$). A distributive lattice is one where the join and meet operators distribute over each other, in other words, a lattice

L is distributive if for all $x, y, z \in L$, $x \otimes (y \oplus z) = (x \otimes y) \oplus (x \otimes z)$ and $x \oplus (y \otimes z) = (x \oplus y) \otimes (x \oplus z)$

The algebraic structure for the Boolean lattice is defined as $\mathcal{B} = (B, \otimes, \oplus, 0, 1, ')$. The unique elements 0 and 1 are the top and bottom concepts necessary for the lattice to be bound, and the complement operator $'$ is defined as above for a complemented lattice. In a finite Boolean algebra, an *atom* is defined as an element $a \in B$ where for any $b \in B$, either $a \otimes b = a$ or $a \otimes b = 0$. The Boolean algebra (and the corresponding lattice) is thus generated from the power set of the atoms (Hirsch and Hodkinson, 2002). As a result, all Boolean algebras with the same number of atoms are isomorphic to each other.

A notion from Boolean algebra that will be used for the modularization is the *ideal*. For a Boolean algebra $\mathcal{B}$ with set of elements B, $I \subseteq B$ is called an ideal in $\mathcal{B}$ if I is nonempty and if for all $i, j \in I$ and $b \in B$ we have $i \otimes b \in I$ and $i \oplus j \in I$. In other words, an ideal is 'closed-downwards' such that it is closed under the lattice meet ($\otimes$) operation.

An ideal is called *proper* if $I \neq \{0\}$ or B. We can also generate an ideal using an element, referred to as the *principal* ideal. If we let B be a Boolean algebra, and $b \in B$, then the principal ideal generated by b is $I(b) = \{a \in B \mid a \leq b\}$.

3.1 Domain Information System

A DIS is composed of three components: an ontology, the data, and an operator which maps data to an ontology. This separation of data from the ontology grants us this ability to manipulate the data through adding or removing records without the need for rechecking consistency or reconstructing the ontology. Figure 1 graphically shows how the three components interact. We first present the ontology.

Definition 3.1 (Abstract Ontology)**.** *Let $\mathcal{C} = (C, \oplus, e_C)$ be a commutative idempotent monoid.*[1] *Let $\mathcal{L} = (L, \sqsubseteq_C)$ be a Boolean lattice, with $L \subseteq C$, such that $e_C \in L$. Let $\mathcal{G} = \{G_{t_i} \overset{\text{def}}{=} (C_i, R_i, t_i)\}_{t_i \in L}$ be a set of rooted graphs at t_i.*

We call an abstract ontology the mathematical structure $O \overset{\text{def}}{=} (\mathcal{C}, \mathcal{L}, \mathcal{G})$.

We recognize the relation on the set of concepts L as the `partOf` relation, denoted by $\sqsubseteq_C$. The corresponding Boolean algebra for this structure is defined as $\mathcal{B} = (L, \otimes, \oplus, e_C, \top, ')$. The binary operators $\otimes$ and $\oplus$ are analogous to the meet and join, but are defined by the relation $\sqsubseteq_C$. The unique elements e_C and $\top$ are the bottom and top concepts of the lattice, and are respectively analogous to 0 and 1. The $\oplus$ operator represents the Cartesian product of the concepts, and it expresses the combination of concepts in the lattice to form new concepts. The ontology is represented as the components within the cloud in Figure 1. The circles represent concepts, and are differentiated by colour to signify whether they are in the Boolean lattice or a rooted graph.

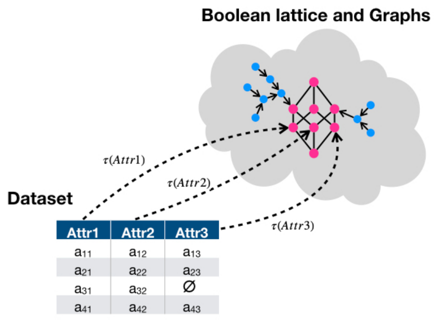


Figure 1: High-level representation of a Domain Information System.

The second component is the data layer, which is formalized using a diagonal-free cylindric algebra. Its operators are indexed with the elements of the carrier set L of the Boolean lattice. In Figure 1, it is represented as the dataset.

Definition 3.2 (Cylindric Algebra)**.** *Let $\mathcal{A} = (A, +, \cdot, -, 0, 1, c_k)_{k \in L}$ be a diagonal-free cylindric algebra (Henkin and Tarski, 1967) such that $(A, +, \cdot, -, 0, 1)$ is a Boolean algebra and c_k is an unary operator on A called cylindrification, and the following postulates are satisfied for any $x, y \in A$, and any $k, \lambda \in L$:*

1. $c_k 0 = 0$
2. $x \leq c_k x$
3. $c_k(x \cdot c_k y) = c_k x \cdot c_k y$
4. $c_k c_\lambda x = c_\lambda c_k x$

We adopt cylindric algebra to reason on data as it allows us to go beyond Relational algebra. Cylindrification operations allow us to handle tuples with undefined values (open world assumption) and we can work on tuples with different length.

Definition 3.3 (Domain Information System)**.** *Let O be an abstract ontology, $\mathcal{A}$ be a diagonal-free cylindric algebra, and a mapping $\tau : A \to L$ as the type operator which relates the set A to elements of the Boolean lattice in O.*

We call a Domain Information System the structure $I = (O, \mathcal{A}, \tau)$.

[1]A monoid is an algebraic structure that has a set S closed under a single associative binary operation $\cdot$, and a distinguished element $e \in S$ called the identity. It is denoted by the triple $(S, \cdot, e)$.

Table 1: Wine Dataset.

Grape	Colour	Sugar	Body	Region	Winery
Merlot	Red	Dry	Full	Niagara	Jackson Triggs
Merlot	Red	Dry	Medium	Okanagan	Jackson Triggs
Pinot Grigio	White	Dry	Medium	Niagara	Konzelmann
Pinot Grigio	White	Semi-sweet	Medium	Niagara	Jackson Triggs
Pinot Blanc	White	Dry	Light	Okanagan	Sperling
Riesling	White	Semi-sweet	Light	Niagara	Jackson Triggs
…	…	…	…	…	…

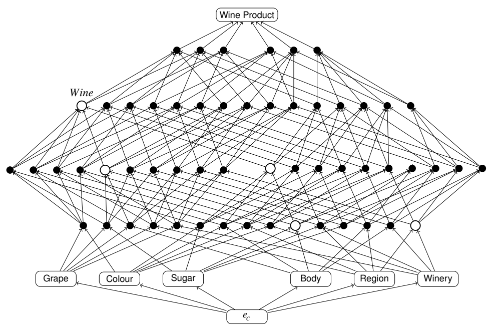


Figure 2: The Boolean lattice for the Wine Ontology.

Figure 1 illustrates this system, with the dataset and ontology linked by the dashed arrows which represents the τ operators. In essence, the abstract ontology is the conceptual level of the information system. The Boolean lattice can be mapped to, using the τ operator, from the data that modeled using the cylindric algebra. In a simplified way, the abstract ontology is analogous to the Terminological Box (T-Box) of DL and the cylindric algebra is the Assertional Box (A-Box).

In Figure 2, we show a Boolean lattice of a DIS representation for the *Wine Ontology* (Noy et al., 2001). The Boolean lattice is constructed from a sample data set shown in Table 1. Each attribute of the data set is a 'part' of the concept *Wine Product*, and thus they are atoms of the Boolean lattice. The remaining concepts are the combinations of these atoms: the power set. Some combinations hold more semantic significance (as determined by domain experts), and are signified by larger hollow nodes. These concepts can be named, such as *Estate* being the combination of *Region* and *Winery*.

3.2 First Isomorphism Theorem

The first isomorphism theorem (Van der Waerden et al., 1950) is used in this work to quantify knowledge loss from modularization.

Theorem 3.1 (First Isomorphism Theorem)**.** *Let R and S be rings, and let $\phi : R \to S$ be a ring homomorphism. Then:*

1. *The kernel of ϕ is an ideal of R,*
2. *The image of ϕ is a subring of S, and*
3. *The image of ϕ is isomorphic to the quotient ring $R/ker(\phi)$.*

In particular, if ϕ is surjective, then S is isomorphic to $R/ker(\phi)$.

The first point introduces the *kernel*, which is a structure associated with the homomorphism. If we define a homomorphism between rings R and S as $\phi : R \to S$, then the kernel is defined as follows:

$$ker(\phi) = \{r \in R \mid \phi(r) = 0_S\} \tag{1}$$

The kernel is used to measure the non-injectivity of the homomorphism. It is important to note any Boolean ring with 1 can be made into a Boolean algebra (Stone, 1936).

4 MODULARIZING THE ONTOLOGY

From Figure 2, it can be seen that a dataset with even a relatively small number of attributes results in a Boolean lattice that is large and possibly unmanageable. The Boolean lattice will be of size 2^n (where n is the number of atoms), thus the number of concepts doubles with each attribute added. Although this can be partially mitigated with clever database design, it minimization cannot be assumed. Using the entire Boolean lattice is both impractical and unreasonable when considering the motivation of this work: before addressing the issues of evolution and data integration, we must employ agents that utilize smaller components of the ontology. Simply using the entire ontology will result in the same tractability issues that exist for existing monolithic ontologies. Thus, sound modularization is necessary to produce views for the agent(s).

Referring to Figure 2, it can be argued that the concept *Wine Product* is composed of two concepts from closely related domains: the *Wine* and *Estate*. The former is composed of *Grape*, *Colour*, *Sugar*, and *Body* whereas the latter is *Region* and *Winery*. For the remainder of this section, we are motivated by being able to modularize the lattice in Figure 2 as a first step of obtaining modules that a *Wine* agent and an *Estate* agent can use for reasoning purposes. That is to say, queries regarding a wine would be delegated to the wine agent, whereas queries about an estate are delegated to the estate agent. Queries that involve elements from both would require the co-operation from both agents. Thus, the modules the agents use need to preserve the knowledge as though they were still a part of the original Boolean lattice (i.e., they are complete).

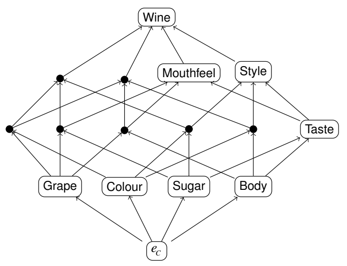


Figure 3: The modularization of the Wine Ontology on *Wine*.

4.1 Generating a Module With View Traversal

This work seeks to generate a module using the View Traversal modularization technique, introduced by Noy and Musen (Noy and Musen, 2009). It is a graphical modularization technique that aims to produce a module that contains all information necessary to answer a query (or set of queries). This technique is formalized for DIS such that it can produce the Boolean sublattice as shown in Figure 3. First, we present the definition of a module so that modularization can be explored:

Definition 4.1 (Module)**.** *Given an abstract ontology $\mathcal{O} = (\mathcal{C}, \mathcal{L}, \mathcal{G})$, a module M of $\mathcal{O}$ is defined as as an abstract ontology $M = (\mathcal{C}_M, \mathcal{L}_M, \mathcal{G}_M)$ satisfying the following conditions:*

- *$C_M \subseteq C$*
- *$\mathcal{L}_M = (L_M, \sqsubseteq_C)$ such that $L_M \subseteq C_M$, $\mathcal{L}_M$ is a Boolean sublattice of $\mathcal{L}$, and $e_c \in L_M$*
- *$\mathcal{G}_M = \{G_n \mid G_n \in \mathcal{G} \otimes t_n \in L_M\}$*

where C_M and C are the carrier sets of $\mathcal{C}_M$ and $\mathcal{C}$, respectively.

In other words, a module is a sub-ontology. This ensures that tasks which can be performed on the ontology (e.g., reasoning) can also be performed on a module. It is evident that abstract ontology given in Figure 3 conforms to the provided definition of a module, if we consider $\mathcal{G}_M = \emptyset$.

View Traversal allows for the freedom to extract a module on any relation to any depth. To ensure that the module we extract conforms to the definition, we restrict the notion of View Traversal on DIS to only traverse the `partOf` relation, and to travel the maximum distance. In the situation that the desired module uses a concept from one of the rooted graphs (i.e., does not use the `partOf` relation), then we first determine the concept that is the root of the graph, then conduct the View Traversal from that concept.

With these restrictions, we claim that the application of View Traversal to a Boolean lattice for a given $c_{\text{st}} \in L$ is equivalent to the principal ideal generated by c_{st}, defined below.

Definition 4.2 (View Traversal Module)**.** *For a given abstract ontology $\mathcal{O} \overset{\text{def}}{=} (\mathcal{C}, \mathcal{L}, \mathcal{G})$ and starting concept $c_{st} \in L$, the module $M_v = (\mathcal{C}_v, \mathcal{L}_v, \mathcal{G}_v)$ is defined as:*

1. *$\mathcal{L}_v = (L_v, \sqsubseteq_C)$ is the principal ideal generated by c_{st}*
2. *$\mathcal{G}_v = \{G_i \mid G_i \in \mathcal{G} \wedge t_i \in L_v\}$*
3. *$C_v = \{c \mid c \in L_{TD} \vee \exists(G_i \mid G_i \in \mathcal{G}_v : c \in C_i)\}$*[2]*.*

where C_v is the carrier set for $\mathcal{C}_v$.

It is possible for a single View Traversal to be created from c_{st}. For example, consider the scenario where we wish to create a single module given two starting concepts $c_1 = (Colour \oplus Sugar)$ and $Taste = (Sugar \oplus Body)$. To achieve this, we are required to extract a single Boolean sublattice that covers both c_1 and $Taste$. As the definition of View Traversal provided uses a single concept, we construct a *proxy starting concept* that is the combination of c_1 and $Taste$, i.e., $c_{\text{st}} = c_1 \oplus c_2$. In more general terms, let C_v be a set of concepts to be modularized on such that $C_v \subseteq L$, then the starting concept for the View Traversal is defined as

$$c_{\text{st}} = \oplus(c \mid c \in C_{\text{V}} : c) \tag{2}$$

Referring to Figure 4, had we taken two View Traversals for c_1 and $Taste$, we would have acquired the two modules shown using the bolded black lines. However, this would provide two distinct modules rather than just one. By determining a proxy c_{st}, we produce a single module that represents the information that could be built using the existing concepts. For example, by considering the proxy, we include the concept *Style* that would have otherwise been forgotten had we taken the union of the two individual modules (as Noy et al. do in (Noy and Musen, 2009)).

With this method, there exists two trivial scenarios: for $c_i, c_j \in C_{\text{V}}$ we have $c_i \sqsubseteq_C c_j$ for disjoint $c_i, c_j \in C_{\text{V}}$, or we have the joins of concepts in C_{V} produce $c_{\text{st}} = \top$. In the former scenario, c_i can be disregarded as according to the definition of the lattice, $c_i \sqsubseteq_C c_j$ implies $c_i \oplus c_j = c_j$. In the latter, the Boolean lattice of the module will be isomorphic to

[2] Throughout this paper, we adopt the uniform linear notation provided by Gries and Schneider in (Gries and Schenider, 1993), as well as Dijkstra and Scholten in (Dijkstra and Scholten, 1990). The general form 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. An empty range is taken to mean true and we write $\star(x \mid: P)$; in this case the range is over all values of variable x.

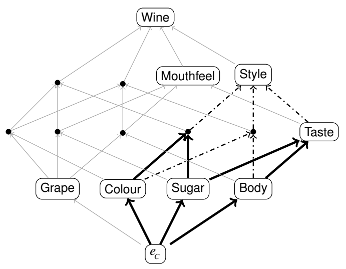


Figure 4: Modularization with $c_1 = Colour \otimes Sugar$ and $Taste = Sugar \otimes Body$.

the original ontology's. Although counterintuitive to modularization, this aligns with the notion of what information can be built. Such a module implies that all atoms of the ontology were required.

We now introduce how View Traversal operates for concepts that belong to one of the rooted graphs of the ontology (i.e., $c_{\text{st}} \in G$). The definition of View Traversal requires $c_{\text{st}} \in L$, thus we define a *rooted* View Traversal into the Boolean lattice as the scenario where $c_{\text{st}} \notin L$. For example, in Figure 5, rooted graphs have been added to the concept *Mouthfeel*. The rest of the Boolean lattice is not shown for easier viewing. These rooted graphs are created by the domain experts at the instantiation of the ontology to enrich the domain by providing additional concepts that are not comprised of the data that is in Table 1. For example, since mouthfeel is `partOf` *Wine*, we can say that a *wine* and *opulence* are related by *associatedWith* relationship, but no data from our domain contains specific data that corresponds to opulent. If there existed data for opulent, it would belong to the Boolean lattice (i.e., our L). With this in mind, we project a View Traversal on the concept *Opulent*, i.e., $c_{\text{st}} = Opulent$. As this concept does not belong to the lattice, we cannot generate the ideal from it. Therefore, we need to search for the root of the graph to which opulent belongs and then work out the View Traversal from it.

Recall that according to the definition of the rooted graphs, the root must be a concept in the Boolean lattice, thus, we can determine the principal ideal of the root. In the example, *Opulent* belongs to a rooted graph which *Mouthfeel* is the root of, thus, *Mouthfeel* would be the *root concept*, and the ideal can be generated as before. Additionally, the set of rooted graphs that belongs to the module will be restricted to only the graphs to which c_{st} belongs. In the example, the graph that contains *Negative* would not be included because it is a different rooted graph.

If both graphs were desired from the View Traversal, the agent would be required to modularize on a concept that is shared among the graphs, such as *Smoothness*. In this scenario, we generate a set of

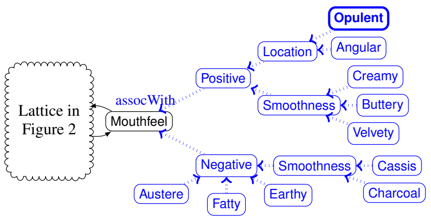


Figure 5: The Wine Ontology enriched with rooted graphs (i.e., $\mathcal{G}$) with root *Mouthfeel*.

root concepts,

$$C_p = \{t_i \mid c_{\text{st}} \in C_i, \text{ where } C_i \text{ is the set of vertices in } G_i\}$$

which is the set of concepts in the Boolean lattice that are the root of a graph that has queried for View Traversal. In other words, for each rooted graph that contains a starting concept for View Traversal, we take its root. Then, with this set of roots, we conduct View Traversal as defined in Equation 2. This results in the following:

Definition 4.3 (Rooted View Traversal). *For a given abstract ontology $O \overset{\text{def}}{=} (\mathcal{C}, \mathcal{L}, \mathcal{G})$ and a set of root concepts C_p corresponding to a $c_{st} \notin L$, the module $M_v = (\mathcal{C}_v, \mathcal{L}_v, \mathcal{G}_v)$ is defined as:*

1. *$\mathcal{L}_v = (L_v, \sqsubseteq_C)$ is the principal ideal generated by $\oplus(c \mid c \in C_p : c)$*
2. $\mathcal{G}_v = \{G_i \mid G_i \in \mathcal{G} \wedge G_i = (C_i, R_i, t_i) \wedge t_i \in L_v \wedge \exists(c_{st} \mid c_{st} \in C_p \wedge c_{st} \in C_i)\}$
3. $\mathcal{C}_v = \{c \mid c \in L_v \wedge \exists(G_i \mid G_i \in \mathcal{G}_v : c \in C_i)\}$

There are two differences between Definitions 4.2 and 4.3. The first is that the Boolean lattice $\mathcal{L}_v$ is generated by the combination of the roots of the graphs in the set C_p rather than the input c_{st}. The second is that not all graphs are included; we only include the graphs that c_{st} is one of its vertices.

4.2 Formalization of View Traversal

The modularization of DIS has been so far defined at the abstract ontology level with Definitions 4.2 and 4.3. We now define the DIS that these abstract ontologies belong to and the function that produces it.

Definition 4.4 (View Traversal). *For a given DIS $I \overset{\text{def}}{=} (O, \mathcal{A}, \tau)$ and concept c such that $c \in \mathcal{C}$, we define a View Traversal as a function such that $VT : I \times c \rightarrow \mathcal{M}_v$, where $\mathcal{M}_v = (O_v, \mathcal{A}_v, \tau_v)$ satisfies:*

1. *O_v is the View Traversal Module generated by c.*
2. $\mathcal{A}_v = (A_v, +, \cdot, -, 0, 1, c_k)_{k \in L_v}$
3. *τ_v is the restriction of τ to only the elements of A_v.*

where L_v is the carrier set of the Boolean lattice $\mathcal{L}_v$ in O_v, and $A_v = \{a \mid a \in A \wedge \tau(a) \in L_v\}$.

We introduce the binary operators of composition and chaining that are carried over View Traversals. We show that, similar to the View Traversal in literature (Noy and Musen, 2009), they are preserved.

Given a DIS $\mathcal{I}$ and two View Traversals T_1 and T_2 that are modularized on starting concepts c_1 and c_2, we define *composition* of View Traversals as both T_1 and T_2 being applied to $\mathcal{I}$:

$$T_1 * T_2 = VT(\mathcal{I}, c_1 \oplus c_2) \tag{3}$$

This definition of composition results in a single module being produced from multiple View Traversals. Additionally, $*$ is both commutative and associative, inheriting the commutativity and associativity of the join operator it utilizes. It is also trivial to show the composition of a View Traversal on c_1 and another View Traversal on c_2 where $c_1 \sqsubseteq_C c_2$ is equal to the View Traversal of c_2:

$$\begin{aligned} & VT(\mathcal{I}, c_1 \oplus c_2) \\ & \iff \langle c_1 \sqsubseteq_C c_2 \Rightarrow c_1 \oplus c_2 = c_2 \rangle \\ & VT(\mathcal{I}, c_2) \end{aligned} \tag{4}$$

The *chaining* of two View Traversals T_1 and T_2 is the process of applying T_2 to the result of T_1:

$$T_2 \circ T_1 = VT(VT(\mathcal{I}, c_1), c_2) \tag{5}$$

This definition of chaining results in the sequential process of 'modularizing a modularization', and is not necessarily commutative. T_2 is restricted to the module produced by T_1, and may no longer include c_2. In this situation, the result would be the empty DIS, i.e., $\mathcal{C} = (\{\}, \{\}, \{\})$.

4.3 Properties of DIS View Traversal

We provide the properties of a module adhering to Definition 4.2. We assume $c_{\text{st}} \neq \top$ to avoid the trivial case of the module being isomorphic to the ontology.

Claim 4.1. *The produced Boolean algebra associated to the Boolean lattice of View Traversal is not a subalgebra of the Boolean algebra associated to the Boolean lattice of the ontology.*

This is easily proven by showing the lack of preservation of operators. In particular, the 0-ary operator of $\top$ is not preserved in the module's associated Boolean algebra.

This claim is significant as the Boolean sublattice (i.e., the module) can be compared to the Boolean lattice of the ontology and be defined as an embedding. However, as it is not a Boolean subalgebra, the algebras cannot be defined in a similar way as an embedding. This distinction results in the ability to guarantee the preservation of the structural information of the concepts when modularizing (i.e., lattice operators of join and meet), but not the information pertaining the additional algebraic operators, summarized in the following claim.

Claim 4.2. *A View Traversal does not preserve the complement operator of the ontology's Boolean lattice.*

This is trivially proven by demonstrating that in the View Traversal, $e'_C = c_{\text{st}}$, but in the ontology, $e'_C = \top$, and due to our earlier stated assumption $c_{\text{st}} \neq \top$, this is a contradiction.

The notions of local correctness and completeness are defined in (d'Aquin et al., 2009), and correspond to the logical modularization techniques that are based on conservative extension. Using these definitions, we make the following claim.

Claim 4.3. *A View Traversal is locally complete but not locally correct with respect to the original ontology.*

The View Traversals completeness follows from the Boolean lattice being an ideal, which by definition, must preserve and be closed under the join and meet operators. However, as shown in Claim 4.2, the complement is not preserved, and thus the module is not locally correct.

5 QUANTIFYING KNOWLEDGE LOSS

An important characteristic of the module is that its Boolean lattice is the principal ideal generated by c_{st} (or the root starting concept c^p_{st}). As a result of the existence of this ideal and the first theorem of isomorphism (Van der Waerden et al., 1950), there must be a homomorphism from one Boolean algebra to another Boolean algebra that this ideal is its kernel.

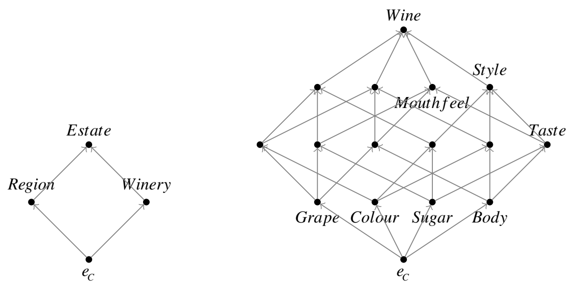
The homomorphism is from one Boolean algebra to another such that $f: B_1 \to B_2$. We define f as follows:

$$f(p) = p \otimes p_0 \tag{6}$$

for every $p \in B_1$ and an arbitrarily chosen $p_0 \in B_1$. We remind the reader that the $\otimes$ operator of a DIS functions identically to the meet operator ($\otimes$) of traditional lattices. The kernel is defined as follows:

$$ker(f) = \{p \in B_1 \mid f(p) = e_C\} \tag{7}$$

To illustrate the utilization of this function, we use the original Boolean lattice found in Figure 2, and use Equation 6, setting $p_0 = Estate$, where $Estate = Region \oplus Winery$. Intuitively, Equation 6 maps every concept from the original Boolean lattice to $Estate$ or



(a) The Boolean lattice mapped to by f (b) The kernel of f

Figure 6: $f(p) = p \otimes Estate$.

one of its parts, populating the kernel with every element that maps to e_C according to Equation 7. As *Estate* and all its parts is a subset of the original Boolean lattice, there will be concepts that map to the same element (i.e., not be injective).

For example, consider the following three evaluations,

$$f(Taste) = Taste \otimes Estate = e_C,$$
$$f(Region) = Region \otimes Estate = Region$$
$$f(Winery) = Winery \otimes Estate = Winery$$

The function maps any value that is a part of *Estate* (such as *Region*) to itself, and anything that has no parts of *Estate* (such as *Taste* or *Grape*) to e_C. With the starting concept $c_{\text{st}} = Estate$, we claim that the kernel of the function f is composed of the concept *Wine* and all its parts, i.e., $ker(f) = \{e_C, Grape, Colour, Sugar, Body, \ldots\}$. Figure 6 displays both Boolean lattices: the one mapped to by f and the one populated by $ker(f)$.

The kernel is specifically the principal ideal generated by the complement of p_0 that we denote by p_0'. In our example, *Wine* is the complement of *Estate* (it is the concept that can be created by combining the atoms that are *not* `partOf` *Estate*). Therefore, the relationship between p_0 and the module from View Traversal can be observed: for any $p_0, c_{\text{st}} \in B$, where $p_0 \neq c_{\text{st}}$, the kernel of f corresponds to the module generated by c_{st}, or more specifically, $VT(c_{\text{st}})$ is equal to the lattice mapped to by f where $p_0 = c_{\text{st}}'$.

The nature of modularizing an ontology into smaller components lends itself to the loss of knowledge. The extraction of a smaller part of the ontology implies *something* is lost. The relationship between the kernel and the Boolean algebra mapped to via the function f embodies this loss of knowledge. If the kernel is taken to be the module extracted via View Traversal, then the Boolean algebra mapped to by f captures the knowledge that is lost. As we associate with the kernel because it is the result of the modularization, we speak of the knowledge that is kept (or lost) in terms of the kernel. More specifically, by assessing and quantifying the size of the kernel, we are measuring what is being lost due to modularization. In Figure 6, the concepts of *Grape*, *Colour*, and any combination of them were lost. Thus, from an ontologist perspective, the manipulation of the homomorphism and the kernel allow for the control of what an agent *will* know (through the kernel) or what an agent *cannot* know (through the Boolean algebra mapped to via f).

One can envision a measure for loss of domain knowledge due to View Traversal proportional to the number of atoms included in the kernel. In other words, for a given View Traversal, VT, we say the percentage of knowledge lost $L(VT)$ is given as

$$L(VT) = \frac{\text{number of atoms in kernel}}{\text{total number of atoms}} \times 100 \qquad (8)$$

We can further study this measure, its scale, and give its laws.

6 DISCUSSION AND FUTURE WORK

As a consequence of the module produced via View Traversal not being locally correct, it implies that what is reasoned from the module may be different from what is reasoned from the ontology. However, since the join and meet operators are preserved in the module, any knowledge deduced in the module using only those operators will be consistent i.e., the same knowledge would be deduced in the original ontology. This is not true for the complement: what they generate from the complement will be true in the context of the module, but will not necessarily be when contextualized by the ontology. For example, referring to Figure 6, the complement of *Region* in the module will be *Winery*. However, in the original ontology (when all atoms are present), the complement of *Region* is not *Winery*, it is the concept c, defined as

$$c = Winery \oplus Grape \oplus Colour \oplus Sugar \oplus Body.$$

We stress that what is generated from the complement (in the context of the module) is merely incomplete: the complement computed in the View Traversal will be a `partOf` the complement computed in the ontology. For example, *Winery* is a part of the true complement c (as is grape, colour, sugar, and body).

Our approach also comes with a direct usage of the algebraic notion of a homomorphism associated to an ideal as a means for measuring the knowledge that is lost due to only focusing on the module. This is extremely important as the many questions that are inherent to modularization need to have an idea to what is left behind when you use a module for reasoning rather than the whole ontology. It is the questions of knowledge coverage by a module and whether a set of modules covers the same knowledge that can be generated using the ontology from which they are obtained. The kernel associated to a module tells us about the concepts that we are omitting through modularization, therefore indirectly telling us about the information and the domain knowledge we are losing. Certainly there is still work to be done on explicitly elaborating this issue. In particular, being able to define loss in more ways than just the size of the kernel. Although this provides us with a way of measuring loss and a way to compare modules based on this, it does not sufficiently capture the idea that there may exist concepts within the ontology that communicate more information or knowledge. As far as we know, we are the first to relate the notion of kernel to the part of knowledge that is lost through modularization. We illustrated this point using View Traversal techniques. In this paper, we simply point to this direction as a viable means for measuring loss of information when using modularization.

7 CONCLUSION

In this paper, we presented the application of View Traversal to an ontology represented with DIS. The utilization of DIS is shown to be advantageous due to the Boolean algebra allowing for the communication of the module as the ideal. The Wine Ontology was utilized as it is a common benchmark for demonstrating a conceptually rich ontology. With this, we are able to formally define what a module is, and how it can be produced using View Traversal on a starting concept, c_{st}. With the homomorphism, it is also possible to create a module that hides a specific set of knowledge. This allows for the extraction of a module based on what an agent *should* or *should not* know. The knowledge that is lost in the modularization process is quantified using the kernel of the homomorphism.

The proposed modularization technique is shown to ensure the completeness of the module. Due to the complement operator, the same cannot be said for the correctness of the module. Extracting a module that is both complete and correct would result in a subalgebra, and correspond to the widely-used techniques that aim to exhibit conservative extension in DL. Additionally, although the kernel quantifies the knowledge that is lost due to modularization, other measurements of knowledge need to be explored. It is common that certain concepts are more significant within a domain, and the loss of such a concept due to modularization should be reflected in the measurement of knowledge loss.

REFERENCES

Algergawy, A., Babalou, S., and König-Ries, B. (2016). A new metric to evaluate ontology modularization. In *SumPre@ ESWC*.

Anand, N., van Duin, R., and Tavasszy, L. (2014). Ontology-based multi-agent system for urban freight transportation. *International Journal of Urban Sciences*, 18(2):133–153.

Ashburner, M., Ball, C. A., Blake, J. A., Botstein, D., Butler, H., Cherry, J. M., Davis, A. P., Dolinski, K., Dwight, S. S., Eppig, J. T., et al. (2000). Gene ontology: tool for the unification of biology. *Nature genetics*, 25(1):25.

Baader, F. (2003). *The description logic handbook: Theory, implementation and applications*. Cambridge university press.

Babalou, S., Kargar, M. J., and Davarpanah, S. H. (2016). Large-scale ontology matching: A review of the literature. In *Web Research (ICWR), 2016 Second International Conference on*, pages 158–165. IEEE.

Bao, J., Voutsadakis, G., Slutzki, G., and Honavar, V. (2009). Package-based description logics. *Modular ontologies*, 5445:349–371.

Belmonte, M.-V., Pérez-de-la Cruz, J.-L., and Triguero, F. (2008). Ontologies and agents for a bus fleet management system. *Expert Systems with Applications*, 34(2):1351–1365.

Benomrane, S., Sellami, Z., and Ayed, M. B. (2016). An ontologist feedback driven ontology evolution with an adaptive multi-agent system. *Advanced Engineering Informatics*, 30(3):337–353.

Bienvenu, M., Kikot, S., Kontchakov, R., Podolskii, V. V., and Zakharyaschev, M. (2018). Ontology-mediated queries: Combined complexity and succinctness of rewritings via circuit complexity. *Journal of the ACM (JACM)*, 65(5):28.

Bodenreider, O. (2004). The unified medical language system (umls): integrating biomedical terminology. *Nucleic acids research*, 32(suppl_1):D267–D270.

Brickley, D. and Miller, L. (2010). Foaf vocabulary specification 0.91.

d'Aquin, M., Schlicht, A., Stuckenschmidt, H., and Sabou, M. (2009). Criteria and evaluation for ontology modularization techniques. *Modular ontologies*, pages 67–89.

Davey, B. A. and Priestley, H. A. (2002). *Introduction to lattices and order*. Cambridge university press.

De Giacomo, G., Lembo, D., Lenzerini, M., Poggi, A., and Rosati, R. (2018). Using ontologies for semantic data integration. In *A Comprehensive Guide Through the Italian Database Research Over the Last 25 Years*, pages 187–202. Springer.

Del Vescovo, C., Parsia, B., Sattler, U., and Schneider, T. (2011). The modular structure of an ontology: Atomic decomposition and module count. In *WoMO*, pages 25–39.

Dietz, J. L. (2006). *What is Enterprise Ontology?* Springer.

Dijkstra, E. and Scholten, C. (1990). *Predicate Calculus and Program Semantics*. Springer-Verlag New York, Inc., New York, NY, USA.

Freitas, A., Panisson, A. R., Hilgert, L., Meneguzzi, F., Vieira, R., and Bordini, R. H. (2017). Applying ontologies to the development and execution of multi-agent systems. In *Web Intelligence*, volume 15, pages 291–302. IOS Press.

Gries, D. and Schenider, F. (1993). *A Logical Approach to Discrete Math*. Springer Texts And Monographs In Computer Science. Springer-Verlag, New York.

Henkin, L. and Tarski, A. (1967). Cylindric algebras.

Hirsch, R. and Hodkinson, I. (2002). *Relation algebras by games*, volume 147. Gulf Professional Publishing.

Huhns, M. N. and Singh, M. P. (1997). Ontologies for agents. *IEEE Internet computing*, 1(6):81–83.

Hustadt, U., Motik, B., and Sattler, U. (2005). Data complexity of reasoning in very expressive description logics. In *IJCAI*, volume 5, pages 466–471.

Jaskolka, J., MacCaull, W., and Khedri, R. (2015). Towards an ontology design architecture. In *Proceedings of the 2015 International Conference on Computational Science and Computational Intelligence*, CSCI 2015, pages 132–135.

Jiménez-Ruiz, E., Kharlamov, E., Zheleznyakov, D., Horrocks, I., Pinkel, C., Skjæveland, M. G., Thorstensen, E., and Mora, J. (2015). Bootox: Practical mapping of rdbs to owl 2. In *International Semantic Web Conference*, pages 113–132. Springer.

Kantamneni, A., Brown, L. E., Parker, G., and Weaver, W. W. (2015). Survey of multi-agent systems for microgrid control. *Engineering applications of artificial intelligence*, 45:192–203.

Khan, Z. C. and Keet, C. M. (2015). Toward a framework for ontology modularity. In *Proceedings of the 2015 Annual Research Conference on South African Institute of Computer Scientists and Information Technologists*, page 24. ACM.

LeClair, A. and Khedri, R. (2016). Conto: a protégé plugin for configuring ontologies. *Procedia Computer Science*, 83:179–186.

Maedche, A., Motik, B., Stojanovic, L., Studer, R., and Volz, R. (2003). Ontologies for enterprise knowledge management. *IEEE Intelligent Systems*, 18(2):26–33.

Marinache, A. (2016). On the structural link between ontologies and organised data sets. Master's thesis.

Motik, B. (2006). *Reasoning in description logics using resolution and deductive databases.* PhD thesis, Karlsruhe Institute of Technology, Germany.

Movaghati, M. A. and Barforoush, A. A. (2016). Modular-based measuring semantic quality of ontology. In *Computer and Knowledge Engineering (ICCKE), 2016 6th International Conference on*, pages 13–18. IEEE.

Nadal, S., Romero, O., Abelló, A., Vassiliadis, P., and Vansummeren, S. (2019). An integration-oriented ontology to govern evolution in big data ecosystems. *Information Systems*, 79:3–19.

Noy, N. and Musen, M. (2009). Traversing ontologies to extract views. *Modular Ontologies*, pages 245–260.

Noy, N. F., McGuinness, D. L., et al. (2001). Ontology development 101: A guide to creating your first ontology.

Pakdeetrakulwong, U., Wongthongtham, P., Siricharoen, W. V., and Khan, N. (2016). An ontology-based multi-agent system for active software engineering ontology. *Mobile Networks and Applications*, 21(1):65–88.

Poggi, A., Lembo, D., Calvanese, D., De Giacomo, G., Lenzerini, M., and Rosati, R. (2008). Linking data to ontologies. In *Journal on data semantics X*, pages 133–173. Springer.

Raimond, Y., Abdallah, S. A., Sandler, M. B., and Giasson, F. (2007). The music ontology. In *ISMIR*, volume 2007, page 8th. Citeseer.

Rector, A., Rogers, J., and Pole, P. (1996). The galen high level ontology.

Sikorski, R., Sikorski, R., Sikorski, R., Mathématicien, P., Sikorski, R., and Mathematician, P. (1969). *Boolean algebras*, volume 2. Springer.

Spackman, K. A., Campbell, K. E., and Côté, R. A. (1997). Snomed rt: a reference terminology for health care. In *Proceedings of the AMIA annual fall symposium*, page 640. American Medical Informatics Association.

Stone, M. H. (1936). The theory of representation for boolean algebras. *Transactions of the American Mathematical Society*, 40(1):37–111.

Suchanek, F. M., Kasneci, G., and Weikum, G. (2008). Yago: A large ontology from wikipedia and wordnet. *Web Semantics: Science, Services and Agents on the World Wide Web*, 6(3):203–217.

Van der Waerden, B. L., Artin, E., and Noether, E. (1950). *Moderne algebra*, volume 31950. Springer.

Wache, H., Voegele, T., Visser, U., Stuckenschmidt, H., Schuster, G., Neumann, H., and Hübner, S. (2001). Ontology-based integration of information-a survey of existing approaches. In *IJCAI-01 workshop: ontologies and information sharing*, volume 2001, pages 108–117. Seattle, USA.

Xiao, G., Calvanese, D., Kontchakov, R., Lembo, D., Poggi, A., Rosati, R., and Zakharyaschev, M. (2018). Ontology-based data access: A survey. IJCAI.

Xue, X. and Tang, Z. (2017). An evolutionary algorithm based ontology matching system. *Journal of Information Hiding and Multimedia Signal Processing*, 8(3):551 – 556. High heterogeneity;Matcher combination;Matching system;Ontology matching;Optimal model;Recall and precision;Semantic correspondence;State of the art;.

Zhou, L., Pan, M., Sikorski, J. J., Garud, S., Aditya, L. K., Kleinelanghorst, M. J., Karimi, I. A., and Kraft, M. (2017). Towards an ontological infrastructure for chemical process simulation and optimization in the context of eco-industrial parks. *Applied Energy*, 204:1284–1298.

Ziegler, P. and Dittrich, K. R. (2004). Three decades of data intecration—all problems solved? In *Building the Information Society*, pages 3–12. Springer.