Formalizing Graphical Modularization Approaches for Ontologies and the Knowledge Loss

Andrew LeClair[1][0000−0003−2779−8000], Ridha Khedri[1], and Alicia Marinache[1]

McMaster University, Hamilton ON, Canada
{leclaial,khedri,marinaam}@mcmaster.ca

Abstract. This paper formalizes the graphical modularization technique view traversal for an ontology component of a Domain Information System (DIS). Our work is motivated by developing the ability to dynamically extract a module (called a view traversal module) based on an initial set of concepts. We show how the ability to quantify the knowledge that is preserved (or lost) in a view traversal module is significant for a multi-agent setting, which is a setting that requires provable privacy. To ensure partial knowledge preservation, we extend the view traversal module to a principal ideal subalgebra module. The cost of this extension is that the obtained knowledge is coarser, as the atoms of the associated lattice are composite yet considered atomic. The presented work constitutes a foundational step towards theories related to reasoning on partial domain knowledge.

Keywords: ontology · ontology modularization · view traversal · view extraction · knowledge loss · provable privacy · autonomous agents.

1 Introduction

Ontology modularization is the process of decomposing an ontology to smaller components for purposes such as reasoning, reusability, and maintenance [60]. In addition to improving a single ontology, the process of modularization can be used to compare and contrast different ontologies [51]. However, despite its wide usage, the process of modularization – and what the precise definition of a module is – is not agreed upon [36]. The lack of a uniform understanding is exemplified when comparing a logical modularization technique [50] to a graphical modularization technique [45]: there are various ways of measuring what a module is, or modules from different techniques may be incomparable to each other. There is a need for having a standard means to assess the quality of an ontology module [2].

Issues such as provable privacy in modules [6, 15], co-ordinating multiple autonomous agents [22], data integration [63], and handling the evolution of the domain [9, 19] could all be served by a formalized modularization process that precisely defines a module. Problem domains regarding whether agents are able to deduce unintended consequences, or how a new data set is to be interpreted by multiple agents all require a formal definition of a module. By establishing what a module is – and what the process of acquiring one is – in a given ontology-based system, we can begin to define what information a module, and by extension, an agent, can have access to.

However, as demonstrated in [42], this is not a trivial set of tasks to solve. It requires a holistic approach to an ontology-based system to discuss how issues such as data integration or autonomous agents can be tackled. This approach is seen with Ontology-based Data Access (OBDA) [47]. OBDA is a system that connects a (possibly heterogeneous) relational dataset to a Description Logic (DL) ontology via mapping. Although OBDA tackles issues such as data integration or evolving concepts, it struggles with multiple agents co-operating [9] or with the evolution of data [59]. Furthermore, OBDA will be limited by the expressivity of the underlying DL fragment [5, 40]. This is an issue when considering that for some ontologies, a DL fragment with lower expressiveness is chosen to accommodate the large amount of data or to avoid certain operators or quantifiers [30]. Agents should be empowered with a system that can express complex concepts rather than be limited due to the amount of data.

This work addresses the problems associated with an evolving domain that is rich with data and has many co-operating agents. To reach this goal, rather than using DL, we use Domain Information System (DIS) [39] to formalize an ontology-based system. The use of DIS allows us to create an optimal ontology-based system, as discussed in [31], as well as one that is easily modularizable to avoid monolithic ontologies [38]. The modularization techniques we introduce are formally defined using the theory of the underlying Boolean algebra. Using the established theory, we address the issues of provable privacy, the lack of a precise definition for a module or modularization, and further, we provide a way to relate logical and graphical modularzation techniques.

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 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

Ontology modularization is an active research field with the aim of ensuring 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 [2, 7, 17, 34, 41, 61]. These approaches vary by the ontology formalism they modularize on, the ontology component(s) 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 [18], but it requires rigorous computation whereas DIS is able to simply compute ideals using its underlying theory.

In [57], the authors highlight the need for the ontology component of an ontology-based system to be modularizable so as to avoid the issues associated with using a monolithic ontology. These issues include the scalability and tractability of reasoning tasks, reusability of components in another problem domain, or the management of the ontology. In addition to the listed problems that arise from a monolithic ontology, in [31], the authors discuss the need to have the ability for several local ontologies to

communicate by using a shared language. DL is the current standard ontology formalism due to its wide usage by research teams and several implementations. However, Wache et al. [57] 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., [33, 46, 62]). These systems require agents to have their own ontology – or pieces of a shared larger ontology – that they can reason on. As the agents are interacting with a smaller more specialized component of the ontology, the queries are more efficiently conducted. 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.

By using techniques from the ontology modularization field, a module can be extracted from an ontology that will be assigned to an agent. 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 [3, 8]. 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 [4, 12, 13, 48, 49, 53, 56]. In addition to this, the lack of an agreed definition of a module results in agents that get possibly inconsisent results to queries [36].

Currently, an ontology-based system will require a full update (recheck for completeness, etc.) whenever an agent's ontology changes due to the domain's evolution [9]. This can be a consuming process depending on how often the agents domain knowledge is expected to change, and can result in a system 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 [47]. The 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. Although existing ontologies can be used, in the case study of [36], it is pointed out that several ontologies are not developed with the intent of being reusable (or easily reused). Thus, often times an ontology engineer will need to create a new ontology for the given data. However, it is not a trivial task to create an ontology from a dataset. The task is described as the bootstrapping problem [32], and OBDA is mostly considered as read-only as it puts restrictions on the ability to modify the datasets and handle updates [59]. Additionally, the query transformation process is not

simple; it depends on multiple aspects of the ontology, queries, and the data consistency [11].

Our research provides 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 (such as reasoning). Additionally, we characterize the modules by the knowledge they do or do not preserve, thus providing a means to communicate provable privacy. We seek to have a single ontology that can be dynamically and easily modularized to avoid a monolithic ontology existing at the higher-level. By extracting the modules 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 with minimal changes to the ontology.

3 Mathematical Background

In this Section, we present the necessary mathematical background for the purpose of making this paper self-contained.

3.1 Lattice Theory and Boolean Algebras

A *lattice* is an abstract structure that can be defined as either a relational or algebraic structure [16]. In our work we use both the algebraic and the relational definitions, there we provide them and we present the connection between them.

Let $(L, \leq)$ be a partially ordered set, we define an upper bound and lower bound as follows. For an arbitrary subset $S \subseteq L$, an element $u \in L$ is called 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. As a relational structure, a lattice is a partially ordered set that is both a join- and meet-semilattice.

A lattice can also be defined as the 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 structures can be connected by the equivalences $a \leq b \iff (a = a \otimes b) \iff (b = a \oplus b)$. The connection between the relational and algebraic definition of a lattice allows us to freely interchange the relational and algebraic aspects in discussion; some concepts are easier to express or explain in one structure over the other.

We 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 has a carrier set (or supporting set) that is a subset of the carrier set of the lattice in which all joins and meets are preserved in the subset.

A Boolean lattice [52] is defined as a *complemented distributive* lattice. A complemented lattice is bounded (i.e., includes a *top* concept ($\top$) and a *bottom* concept ($\bot$)), and where every element a has a *complement* (i.e., 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, i.e., 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 that bound the lattice, that we denoted above as $\bot$ and $\top$ respectively, 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 $a \otimes b = 0$ [26]. In this work, we only consider finite Boolean algebras. The Boolean lattice and its corresponding Boolean algebra is thus generated from the power set of the atoms [29]. As a result, all Boolean algebras with the same number of atoms are isomorphic to each other.

Two distinguished types of substructures are an *ideal* and *filter*. For a Boolean algebra $\mathcal{B}$ with the carrier set 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 $i \oplus j \in I$. A filter is the dual of an ideal. For a Boolean algebra $\mathcal{B}$ with set of elements B, $F \subseteq B$ is called a filter in $\mathcal{B}$ if F is nonempty and if for all $i, j \in F$ and $b \in B$ we have $i \oplus b \in F$ and $i \otimes j \in F$

An ideal is called *proper* if $I \neq \{0\}$ or B. It is possible to generate an ideal using an element, referred to as the *principal* ideal. Let $\mathcal{B}$ be a Boolean algebra, and $b \in B$, then the principal ideal generated by b is $I(b) = \{a \in B \mid a \leq b\}$. In this paper, we use the symbol $L_{\downarrow b}$ to denote the set of elements in the principal ideal generated by an element b. A filter is defined as the dual to the ideal. For the principal filter generated by an element, we use a symbol with a similar definition to the one used for a principal ideal, but for a principal filter instead: $L_{\uparrow c_{\text{st}}}$.

An ideal is *maximal* (or a *prime* ideal) if $I \neq B$ and the only proper ideal containing I is B itself. The dual of of a maximal ideal is the *ultrafilter*. It is also established that for any maximal ideal (or ultrafilter) $I \subseteq B$ and any element $x \in B$, I contains exactly one of $\{x, x'\}$.

3.2 Domain Information System

A Domain Information System (DIS) [39] consists of three components: a domain ontology, a domain data view, and a function that maps the two. This separation of data from the ontology grants us the ability to manipulate the data by adding or removing records without the need for rechecking consistency or reconstructing the ontology. Figure 1 shows these three components. Let C be a set of concepts, e_c an element of C such that $\forall(c \mid c \in C : c \oplus e_c = e_c \oplus c = c)$. Moreover, $\oplus$ is associative, idempotent, and commutative. Hence, $(C, \oplus, e_C)$ is a commutative, idempotent monoid[1].

Let $C_i \subseteq C, R_i \subseteq C_i \times C_i$, and $t \in C_i$. 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: $t \in C_i$ is *root* of $G_i^t \iff \forall(k \mid k \in C_i : k = t \vee (k,t) \in {R_i}^+)$.

With this, we present the definition of the ontology.

[1] A monoid is an algebraic structure with a single associative operator, and an identity element.

Definition 1 (Domain Ontology). *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* $\mathcal{G} = \{G_i^t\}_{i \in I,\ t \in L}$ *a set of rooted graphs at* t*.*

A domain ontology is the mathematical structure $O \stackrel{\text{def}}{=} \big(\mathcal{C}, \mathcal{L}, \mathcal{G}\big)$*.*

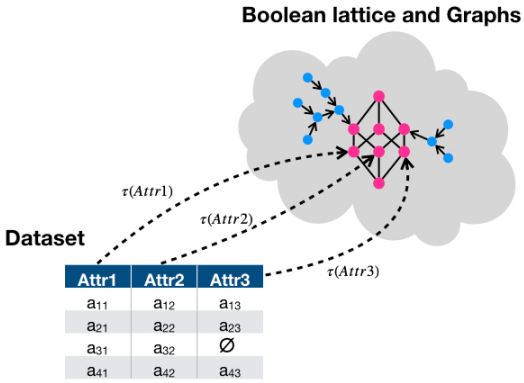


Fig. 1: High-level representation of a Domain Information System

We recognize the relation on the set of concepts L as the `partOf` relation, denoted by $\sqsubseteq_C$. The corresponding Boolean algebra for the Boolean lattice of a domain ontology 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 related to $\sqsubseteq_C$ by $a \oplus b = b \iff a \sqsubseteq_C b \iff a \otimes b = a$. The unique elements e_C and $\top$ are the bottom and top concepts of the lattice, and are respectively analogous to 0 and 1.

A concept $k \in C$ is called *atomic* if it has no sub-parts, i.e.,: k is atomic $\stackrel{\text{def}}{\iff} \forall (k' \mid k' \in L : k' \sqsubseteq_C k \implies k' = k \vee k' = e_C)$.

In Figure 1, the ontology is represented as the components within the cloud. The circles represent concepts, and are differentiated by colour to signify whether they are in the Boolean lattice (i.e., pink) or a rooted graph (i.e., blue).

The second component is the domain data view associated with an ontology $\big(\mathcal{C}, \mathcal{L}, \mathcal{G}\big)$, which is formalized using a diagonal-free cylindric algebra. Diagonal-free cylindric algebra has been introduced in [28]. Its cylindrification operators are indexed over the elements of the carrier set L of the Boolean lattice. In Figure 1, it is represented as the dataset.

Definition 2 (Domain Data View Associated to an Ontology). *Let* $O = \big(\mathcal{C}, \mathcal{L}, \mathcal{G}\big)$ *be a domain ontology as defined in Definition 1, where* L *is the carrier set of* $\mathcal{L}$*. A domain data view associated to* O *is the structure* $\mathcal{A} = (A, +, *, -, 0_A, 1_A, \{c_\kappa\})_{\kappa \in L}$ *that is a diagonal-free cylindric algebra 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. $c_\kappa 0 = 0$
2. $x \leq c_\kappa x$
3. $c_\kappa(x * c_\kappa y) = c_\kappa x * c_\kappa y$
4. $c_\kappa c_\lambda x = c_\lambda c_\kappa x$

We adopt cylindric algebra to reason on data as it gives us a data view that goes beyond the relational view of Codd [14]. Cylindrification operations allow us to handle tuples with undefined values (an open world assumption) and we can work on tuples with different length. Ultimatley, $\mathcal{A}$ gives us the data view. With both of these components, a DIS can be defined as follows.

Definition 3 (Domain Information System). *Let O be a domain ontology, $\mathcal{A}$ be its domain data view, and a mapping $\tau : A \rightarrow L$ as the operator which relates the set A to elements of the Boolean lattice in O.*

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

Figure 1 illustrates this system, with the dataset and ontology linked by the dashed arrows that represents the τ operator. In essence, the domain ontology is the conceptual level of the information system.

In Figure 2, we show a Boolean lattice of a DIS representation for the *Wine Ontology* [44]. The Boolean lattice is constructed from a sample data set shown in Table 1. Each attribute of the data set is a `partOf` the concept *Wine Product*, and they are the atoms of the Boolean lattice. The remaining concepts are the combinations of these atoms, formed from the power set of the Boolean lattice atoms. Some combinations hold more domain relevance (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*, while others may not have explicit names in the domain and instead be referred to by the combination of its parts (e.g., *Grape* $\oplus$ *Colour*).

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
...	...	...	...	...	...

3.3 Conservative Extension, Local Correctness, and Local Completeness

A prevalent form of modularization (referred to as 'logical modularization techniques') defines a module as a set of concepts and relations such that the knowledge that can

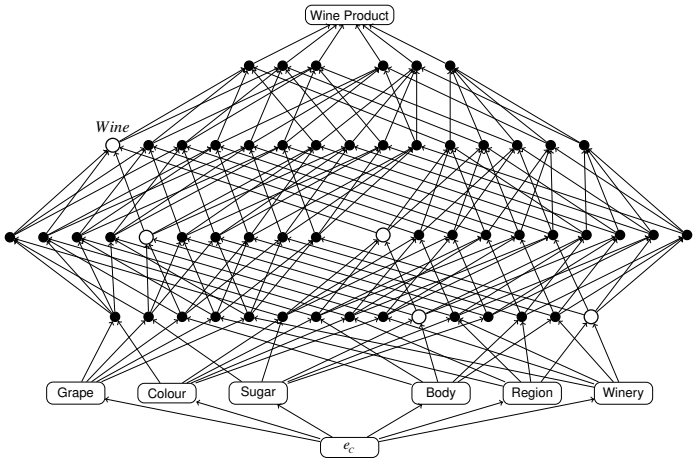


Fig. 2: The Boolean lattice for the Wine Ontology [35]

be deduced from these concepts is preserved in the module. This presevation of the knowledge is determined by extracting a module such that the ontology conservatively extends it.

Conservative extension is a notion from mathematical logic, often used in fields such as proof theory. In [37], we find the (proof theoretic) conservative extension is presented as follows:

Definition 4 ([37]). *Let L and L' be logics. We call L' a (proof theoretic) conservative extension of L provided that all formulas of L are formulas of L' and that, for all formulas A of L, A is a theorem of L iff A is a theorem of L'.*

In addition to the definition provided, there exists a stronger notion of conservative extension called *model theoretic* conservative extension. It is presented as follows:

Definition 5 ([37]). *Let T and T' be theories. We call T' a (model theoretic) conservative extension of T if every model of T can be extended to a model of T'.*

As seen in [37], both types of conservative extension are used to define an ontological module. In the context of DL, the formulas are the Terminological Box (T-Box) sentences. When demonstrating the conservative extension of the ontology to the module, it is common that it is shown in two parts, such as in [24]. The first part, referred to as *local correctness*, shows that all formulas of L are formulas of L'. The second part, referred to as *local completeness*, shows that for all formulas A of L, A is a theorem of L iff A is a theorem of L'.

3.4 First Isomorphism Theorem

The first isomorphism theorem [58] is used in this work to quantify knowledge loss from modularization.

Theorem 1 (First Isomorphism Theorem [58]). *Let R and S be rings, and let $\phi : R \to S$ be a ring homomorphism. Then the kernel of ϕ is an ideal of R. In particular, if ϕ is surjective, then S is isomorphic to $R/ker(\phi)$.*

This theorem introduces the *kernel*, which is a structure associated with the homomorphism. Let R and S be two rings. Then for a homomorphism $\phi : R \to S$, 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 the 0-ary constant 1 can be made into a Boolean algebra [54].

4 Modularizing The Ontology

The goal of modularization is to improve the reusability and maintainability of the ontology, as well as make tasks such as reasoning more tractable. In addition to this, the process of modularization can address other issues, such as provable privacy or conceptualizing a heterogeneous system of multiple agents co-operating. In this section, we introduce two modularization techniques as well as the properties of the produced modules.

Figure 2 shows a domain ontology that has been conceptualized for a Wine dataset. Although seemingly large, this is before any processing has been done to improve the maintainability or reusability. For instance, 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*. In a multi-agent system, it might be more suitable for this ontology to be modularized such that one agent has the module (and the knowledge associated to the concepts therein) for *Wine*, and a different agent has the module for *Estate*. It could also be the case that the agent that has the knowledge about *Wine* must not have knowledge of the *Estate* (and vice versa). For the remainder of this section, we will introduce two modularization techniques: one based on view traversal, and another based on the principal ideal subalgebra. We will then show how the modularization challenges described above are addressed with these modules.

4.1 The Determination of a Graphical Modularization Technique

In the field of ontology modularization, there are several implementations of graphical modularization techniques. This includes Prompt [45], Magpie [21], SWOOP [23], ModOnto [10], PATO [55], and SeeCOnt [1]. We consider these techniques in order to draw parallels between our DIS-based graphical modularization techniques and those that already exist.

Rather than extract a single module, both PATO and SeeCOnt implement techniques that instead partition the ontology. These methods seek to partition an ontology into a (possibly disjoint) set of modules such that every concept belongs to a module. In this

paper, we are concerned with extracting a module that meets a specific requirement for an agent. For this reason, we do not consider these two techniques further.

From the remaining techniques, we observe a prominent usage of the view traversal technique or a derivative of it. Prompt is the tool developed as the implementation of view traversal, therefore it is natural that it is the most true to the developed algorithm. Magpie implements a technique that operates similar to view traversal, however, rather than indiscriminately including all traversed concepts, it only retains the ones necessary for the preservation of axioms or consequences of the input. SWOOP is discussed as being able to faciliate a modularization process that factors an ontology into 'subsidiary domains', which are analogous to a view from view traversal. Finally, ModOnto operates similar to Magpie in that it includes concepts necessary for the preservation of axioms and consequences.

All of the presented modularization techniques can be traced to operate similar to view traversal: given an input query, a set of concepts and relationships are produced to answer that query. The techniques vary in how they deem a concept to be eligible for the module or not, but all share this same goal of answering a query with a 'view'. For this reason, our presented graphical modularization technique operates the same. By doing this, the presented technique can be considered an implementation of these techniques.

4.2 Modularization of DIS

To modularize a DIS, a formalized understanding of what a module – and its properties – must be introduced. Thus, we first write a formal definition of a module, and in the context of a DIS, it is defined as follows:

Definition 6 (Module). *Given a domain ontology $O = (\mathcal{C}, \mathcal{L}, \mathcal{G})$, a module M of O is defined as as a domain 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* $e_c \in L_M$
- $\mathcal{G}_M = \{G_n \mid G_n \in \mathcal{G} \wedge t_n \in L_M\}$

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

In short, a module is a sub-ontology. This ensures that the tasks that can be performed on an ontology – such as reasoning or modularization – can also be performed on a module. As the concepts of the module are a subset of the ontology's concepts, we achieve the first goal.

Let $\mathbb{O}$ be a set of ontologies. The *modularization* is a function

$$M : \mathbb{O} \to \mathbb{O}$$

such that $M(O) = O_M$ where O_M is a module of O.

The central piece of a module (and thus, modularization) is the Boolean lattice. More specifically, the process of modularization is the process of determining a Boolean sublattice that will construct the module. With this, we understand the modularization techniques as transformations to the underlying Boolean algebra.

4.3 Generating a Module With View Traversal

A key component of view traversal is the declaration of a concept (or set of concepts), referred to as the starting concept c_{st}, that the module should be focused on. From these starting concepts, the ontology is traversed to acquire the concepts (and the relations) that will be used to populate the module. In a DIS, we define a view traversal using a principal ideal generated by c_{st}, written as $L_{\downarrow c_{\text{st}}}$. The principal ideal is a Boolean sublattice defined with a set of concepts such that it contains every concept that is `partOf` c_{st}. Figure 3 shows a module that has been generated using the starting concept *Wine*.

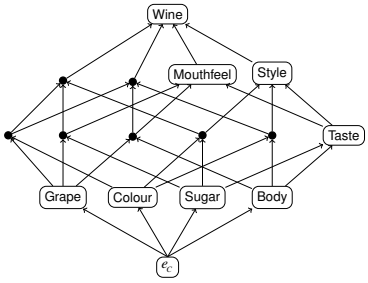


Fig. 3: The Boolean lattice of the module extracted from the Wine Ontology with $c_{\text{st}} =$ *Wine* [35]

As the principal ideal is generated by the input c_{st}, it is critical that c_{st} is well-defined. We envision three cases related to the concept c_{st} under consideration:

1. The starting concept is in the Boolean lattice
2. The starting concept is in a rooted graph
3. There are several starting concepts

The first case is the business-as-usual case, and operates as defined above. In the second case, the starting concept is determined using a *projected concept*, c_p. This projected concept is the root of the rooted graph that contains the desired concept. For example, referring to Figure 4, if $c_{\text{st}} = \mathit{Location}$, then the projected concept would be the root of the graph, *Mouthfeel*. Thus, the principal ideal would be generated using *Mouthfeel*. Finally, in the third scenario where there are multiple desired starting concepts, we denote by C_{V} the set $C_{\text{V}} \subseteq L$. It is the set of all desired starting concepts or their projected concepts. The starting concept is then defined as

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

[2] Throughout this paper, we adopt the uniform linear notation provided by Gries and Schneider in [25], as well as Dijkstra and Scholten in [20]. 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.

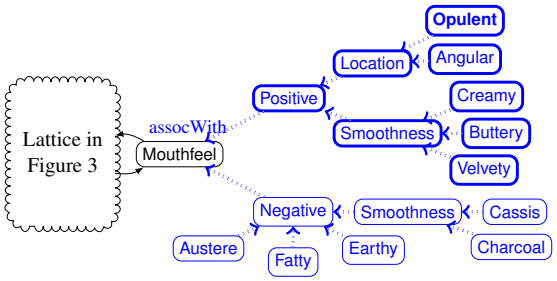


Fig. 4: The Wine Ontology enriched with rooted graphs (i.e., $\mathcal{G}$) with root *Mouthfeel* [35]

For example, referring to Figures 3 and 4, consider that we wish to modularize based on the starting concepts *Opulent*, *Mouthfeel*, and *Taste*; we first determine all projected concepts. In this case, there is only one concept that belongs to a rooted graph: *Opulent*. The root for this graph is *Mouthfeel*, and so this is taken as the projected concept. Following this, we determine the composition of the three concepts together: $c_{\text{st}} = \mathit{Mouthfeel} \oplus \mathit{Mouthfeel} \oplus \mathit{Taste}$. The result is *Mouthfeel*, thus it is the starting concept for view traversal.

In Figure 5 we present a module that is formed from a starting concept that is the composition of two elements, and the composition is not one of the two concepts. Using starting concepts $\mathit{Colour} \oplus \mathit{Sugar}$ and *Taste*, we would extract the two modules shown with the respective tops, denoted by bold lines. Simply taking these two modules, would be missing information that could be made from concepts that belong to the two modules: the combination of *Colour* with *Body*, as well as the combination of all three atoms. Therefore, taking a module with the concept that results from Equation (2), we instead get the module that contains those concepts connected via bold lines, and also via dashed lines. This module includes the two stated concepts, as well as those from the two original modules.

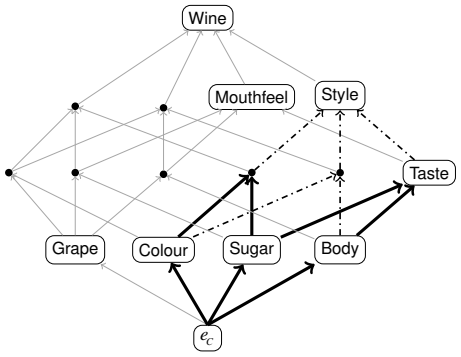


Fig. 5: Modularization with $c_1 = \mathit{Colour} \oplus \mathit{Sugar}$ and $\mathit{Taste} = \mathit{Sugar} \oplus \mathit{Body}$ [35]

With this approach, there exist two trivial cases: the first is for $c_i, c_j \in C_{\mathrm{V}}$ we have $c_i \sqsubseteq_C c_j$ for any $c_i, c_j \in C_{\mathrm{V}}$, and the second is the composition of concepts in C_{V} produce $c_{\mathrm{st}} = \top$. In the former, c_i can be disregarded as $c_i \sqsubseteq_C c_j$ implies $c_i \oplus c_j = c_j$. In the latter, the Boolean lattice of the module will be the original ontology. Although counterintuitive to modularization – as the module will not be any smaller than the ontology – this aligns with the notion of a module containing the concepts necessary to answer the input query. Such a module implies that all atoms of the ontology were required to answer the query, and therefore, every concept in the lattice of the ontology can be formed via a combination of atoms.

For the remainder of this paper, when considering c_{st}, we assume it to be normalized such that c_{st} is a single concept within the Boolean lattice. We now present the definition of view traversal for an ontology within DIS, as well as a view traversal module.

Definition 7 (View Traversal). *Let $\mathcal{O} = (\mathcal{C}, \mathcal{L}, \mathcal{G})$ be an ontology. We also let $\mathbb{M}_D$ be the set of all possible view traversal modules obtainable from $\mathcal{O}$. view traversal vt_D is a function that is declared as follows*

$$vt_D : \mathbb{M}_D \times C \to \mathbb{M}_D \tag{3}$$

Where $vt_D(\mathcal{O}, c_{st}) = (\mathcal{C}_v, \mathcal{L}_v, \mathcal{G}_v)$ and is defined as:

1. $\mathcal{L}_v = (L_{\downarrow c_{st}}, \sqsubseteq_C)$ *is a Boolean lattice*
2. $\mathcal{G}_v = \{G_i \mid G_i \in \mathcal{G} \wedge t_i \in L_v\}$
3. $C_v = \{c \mid c \in L_{\downarrow c_{st}} \vee \exists (G_i \mid G_i \in \mathcal{G}_v : c \in C_i)\}$.

Given a view traversal module and a starting concept, the process of applying view traversal will produce a new view traversal module. It is important to note that the original domain ontology belongs to the set of all possible view traversal modules, i.e., $\mathcal{O} \in \mathbb{M}_D$, as by setting c_{st} to be $\top$, a view traversal module is produced that is equal to $\mathcal{O}$. This, and other trivial scenarios for view traversal, are expressed in the following Lemma. First we introduce the empty ontology and atomic ontology, denoted as $\mathcal{O}_0$ and $\mathcal{O}_a$, respectively. The empty ontology is an ontology with the carrier set being only the empty concept. The atomic ontology is an ontology with the carrier set of the Boolean lattice being two elements: an atom a and the empty concept. Unlike $\mathcal{O}_0$ which characterizes both C and L to be the single element e_C, $\mathcal{O}_a$ only characterizes L as there may still be rooted graphs (with root a) that have concepts populating C.

Lemma 1. *Given a domain ontology $\mathcal{O}$, then the following is true:*

1. $vt_D(\mathcal{O}, \top) = \mathcal{O}$
2. $vt_D(\mathcal{O}, \bot) = \mathcal{O}_0$
3. $vt_D(\mathcal{O}, a) = \mathcal{O}_a$ *for any atom a in L*
4. $\forall (c_1, c_2 \mid c_1, c_2 \in C : (c_1 \not\sqsubseteq_C c_2 \wedge vt_D(\mathcal{O}, c_2) = (\mathcal{C}_v, \mathcal{L}_v, \mathcal{G}_v)) \implies c_1 \notin C_v)$

Similar to Noy and Musen in [43], we define the analogue to the *composition* and *chaining* view traversals.

Claim (View Traversal Composition). Let $\mathcal{O} = (\mathcal{C}, \mathcal{L}, \mathcal{G})$ be an ontology. Let c_1, c_2 be two concepts in the carrier set of $\mathcal{L}$ such that $c_2 \sqsubseteq_C c_1$.

$$vt_D(vt_D(O,c_1),c_2) = vt_D(O,c_2)$$

Proof. A proof by contradiction. Let O_1 be the result of $vt_D(M_v,c_1)$. Assume that there exists a concept c that belongs to the carrier set of $vt_D(M_v,c_2)$ but does not belong to the carrier set of $vt_D(O_1,c_2)$. As the carrier set of O_1 includes all parts of c_1 as per the definition of a view traversal module, this implies that c is a part of c_2 but is not a part of c_1. Due to the transitivity of the relation, this results in a contradiction that $c \sqsubseteq_C c_2$ and $c_2 \sqsubseteq_C c_1$ but not $c \sqsubseteq_C c_1$.

In essence, view traversal composition is the act of 'modularizing a modularization'. In the situation that c_2 is not a part of c_1, the result is O_0. This is because $vt_D(O,c_1)$ will not contain c_2. Therefore, to take a module on a concept that does not exist (to the ontology being modularized) should result in 'nothing', which for us is the empty ontology.

The second function introduced is the chaining of two view traversal modules. The chaining of two view traversal modules is the act of combining two modules into a single module, and is defined as follows.

Claim (View Traversal Combination). Let M_1 and M_2 be any view traversal modules from $\mathbb{M}_D$ for a given DIS, $\mathcal{D} = (\mathcal{O},\mathcal{A},\tau)$. Let $*$ be an operator with the signature

$$* : \mathbb{M}_D \times \mathbb{M}_D \to \mathbb{M}_D$$

We have $vt(M_1,c_1) * vt(M_2,c_2) = vt(O, c_1 \oplus c_2)$.

The view traversal combination function inherits the commutativity and associativity from $\oplus$. It is also trivial to show the view traversal combination on view traversals with tops c_1 and c_2 where $c_1 \sqsubseteq_C c_2$ is equal to the view traversal of c_2 as $c_1 \sqsubseteq_C c_2 \Rightarrow c_1 \oplus c_2 = c_2$.

4.4 Properties of DIS View Traversal

We now provide the following proposition which lists a series of claims for a module adhering to Definition 7. We assume $c_{\text{st}} \neq \top$ to avoid the trivial case of the module being equal to the original ontology.

Proposition 1. *Let $O = (\mathcal{C},\mathcal{L},\mathcal{G})$ be an ontology and $O_v = (\mathcal{C}_v,\mathcal{L}_v,\mathcal{G}_v)$ be a module obtained from O using view traversal. Then:*

1. *The algebra associated to $\mathcal{L}_v$ is not a subalgebra of that associated to $\mathcal{L}$.*
2. *The Boolean algebra associated with the ontology that is produced from view traversal does not preserve the complement operator of the Boolean algebra associated with the original ontology.*
3. *The view traversal module is locally complete but not locally correct with respect to the original ontology.*

The proof for the above results can be found in [35]. Proposition 1.1 indicates that although the Boolean lattice of the view traversal module is a sublattice of the Boolean lattice of the original ontology, the Boolean algebra associated with the view traversal

module is not a subalgebra of the Boolean algebra associated with the original ontology. Proposition 1.2 states that specifically, the complement operator is not preserved from the Boolean algebra associated with the original ontology. Proposition 1.3 indicates that due to the inability to preserve the complement operator of the Boolean algebra associated with the original ontology, the module is not locally correct, as defined in [37].

4.5 The Principal Ideal Subalgebra Module

The Boolean lattice of the module produced from view traversal is a Boolean sublattice of $\mathcal{L}$ but the Boolean algebra associated to the module is not a Boolean subalgebra of the algebra associated to $\mathcal{L}$. This may be satisfactory from a pragmatic approach as the resulting module can be utilized in the same ways as an ontology (e.g., reasoning, modularization). However, from a more rigorous perspective, we require an ability to produce a module that preserves the knowledge from the ontology. Thus, we require a module that in addition to preserving the join and meet opertors, also preserves the complement operator, i.e., is a Boolean subalgebra.

The view traversal module is ultimately formed using the principal ideal on a starting concept c_{st}. It is easy to show that the principal ideal generated by c_{st} unioned with the principal filter generated by c'_{st} – the complement of c_{st} – is a Boolean subalgebra of the Boolean algebra associated with the original ontology [27]. Let $c_{\text{st}} \in L$ and c'_{st} be its complement. Let $M_v = vt_D(O, c_{\text{st}}) = (\mathcal{C}_v, \mathcal{L}_v, \mathcal{G}_v)$ and let $\mathcal{B}$ be the Boolean algebra associated to the lattice of O. We indicated that $\mathcal{L}_{\sqsubseteq}$ is a principal ideal in $B1$, with the carrier set $L_{\downarrow c_{\text{st}}}$. We denote by $L_{\uparrow c'_{\text{st}}}$ the carrier set of the principal filter generated by c'_{st}. Then the set $L_{\uparrow c'_{\text{st}}} \cup L_{\downarrow c_{\text{st}}}$ gives us the carrier set for a Boolean subalgebra of $\mathcal{B}$. We call the module that is formed using this set as the *principal ideal subalgebra module* and is formally defined below.

Definition 8 (Principal Ideal Subalgebra Module). *For a given domain ontology* $O \stackrel{\text{def}}{=} (\mathcal{C}, \mathcal{L}, \mathcal{G})$ *and starting concept* c_{st} *in the carrier set* L *of* $\mathcal{L}$*, we define the set* $L_P = L_{\downarrow c_{st}} \cup L_{\uparrow c'_{st}}$*. The module* $M_P = (\mathcal{C}_P, \mathcal{L}_P, \mathcal{G}_P)$ *is defined as:*

1. $\mathcal{L}_P = (L_P, \sqsubseteq_C)$ *is a Boolean lattice*
2. $\mathcal{G}_P = \{G_i \mid G_i \in \mathcal{G} \wedge t_i \in L_P\}$
3. $\mathcal{C}_P = \{c \mid c \in L_P \vee \exists (G_i \mid G_i \in \mathcal{G}_P : c \in C_i)\}$

where C_P *is the carrier set for* $\mathcal{C}_P$*.*

We present an example of the principal ideal subalgebra module in Figures 6 and 7. Figure 6 shows the Boolean lattice for the Wine ontology with the concepts that would populate the carrier set of the principal ideal subalgebra if $c_{\text{st}} = Grape \oplus Sugar \oplus Body$. The concept c_{st} corresponds to the concept named *Mouthfeel* in the Boolean lattice of the original ontology. Two disjoint sublattices can be seen. The first one is the magenta principal ideal that is formed by *Mouthfeel*, which contains the empty concept. The second one is the cyan principal filter that is formed by the complement of *Mouthfeel* and which contains the top concept. By extracting these concepts and constructing a

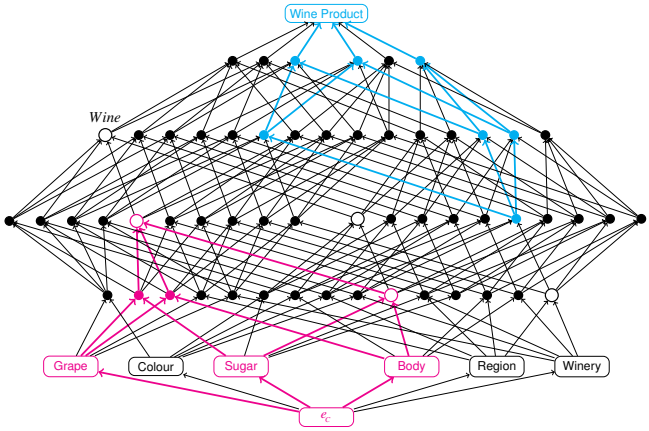


Fig. 6: The Boolean lattice for the Wine Ontology with the principal ideal subalgebra module highlighted for $c_{\text{st}} = Mouthfeel$ [35]

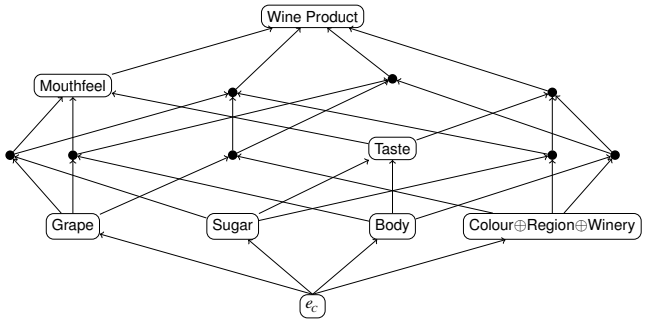


Fig. 7: The Boolean lattice of the principal ideal subalgebra module extracted from the Wine Ontology with $c_{\text{st}} = Mouthfeel$ [35]

new Boolean lattice, we result with the structure shown in Figure 7. The Boolean lattice in Figure 7 has a carrier set formed from the union of the principal ideal generated by c_{st} and the principal filter generated by c'_{st}. We note that an atom in this module, the concept $Colour \oplus Region \oplus Winery$, was not an atom in the original ontology. This is because by producing a module in this manner, we effectively change our understanding of what the atomic blocks of the domain are. In the module, the concept $Colour \oplus Region \oplus Winery$ relates to data that is non-decomposable and atomic in Figure 7, whereas in the original ontology, we know that this same concept is composed of the the concepts *Colour*, *Region*, and *Winery*. We refer to this as the *coarsening* of the domain, and therefore the knowledge that would be derived from this module would be *coarser* than the knowledge that would be derived from the original ontology.

By extending the view traversal module with the principal filter generated by c'_{st}, we remedy the inability to preserve the complement. Thus, we have the following claim.

Claim. A module obtained by extending view traversal with the principal filter generated with c'_{st} is locally correct and locally complete.

The above claim holds by construction. The module is formed using a Boolean subalgebra, which is conservatively extended by the ontology's Boolean algebra.

According to this result, every satisfiable formula in the module is satisfiable in the ontology, and every satisfiable formula in the ontology written in the language of the module is satisfiable in the module.

The relationship between the two modules in Definitions 7 and 8 can be seen in the underlying Boolean lattices. Specifically, any view traversal module can be *extended* to a principal ideal subalgebra module by extending the carrier set of the Boolean lattice of the module with the filter generated by c'_{st} (and the related rooted graphs). Likewise, a principal ideal subalgebra can be *truncated* to a view traversal module by removing any element belonging to the principal filter generated by c'_{st} (and the related rooted graphs) from the carrier set of the Boolean lattice.

The inclusion of additional concepts to the module raises the concerns of whether the modularization process will be fulfilling the requirements that a module should be smaller in size. It must be easy to determine if the process of modularization will produce a module that is associated with a Boolean subalgebra that is strictly embedded in the original Boolean algebra.

Claim (Maximal Principal Ideal Subalgebra Module). Let c_{st} be a concept in the ontology's Boolean lattice L. If the principal ideal generated by c_{st} is a maximal ideal, then the resulting principal ideal subalgebra module will be equal to the original ontology.

Proof. This follows from the definitions of a maximal ideal and ultrafilter. More specifically, for any $x \in L$, a maximal ideal I will contain exactly one of x or x'. As the corresponding ultrafilter, F, is the dual of I, then for every $x \in I$, x' must belong to F. Therefore, every element from L will be in either I or F.

Claim (Non-isomorphism of Module). Let c_{st} be a concept in the Boolean lattice L. If the principal ideal generated by c_{st} is not a maximal ideal, then the resulting principal ideal subalgebra will not be isomorphic to the original Boolean algebra.

Proof. We prove this by contradiction. We assume that the principal ideal generated by c_{st} is not maximal. We also assume that every element $x \in L$ belongs to either the principal ideal I, or the corresponding filter F. This implies every atom a belongs to either I or F. For the principal ideal to be proper, it cannot include every atom in its set. Similarly, for the filter to be proper, it can at most contain one atom. Therefore, I must contain every atom save for one which must belong to F. An ideal that contains all atoms save one is a maximal ideal (and a filter generated using an atom is an ultrafilter), contradicting the assumption they are not maximal.

As a result of the final claim, a principal ideal subalgebra module generated using any concept that is directly related to the top concept with the `partOf` relation (i.e., a concept that is the combination of all but one atoms) is guaranteed to be smaller in size than the original ontology.

A result of the module being formed using the principal ideal and a principal filter is that the bottom concept of the principal filter (i.e., c'_{st}) will be an atom of the module's lattice. However, there is no restriction that the principal filter be generated using an atom. In fact, expressed in the proof of the final claim, if the module is non-maximal (i.e., it is not isomorphic to the original ontology) then c'_{st} is not an atom of the ontology. Therefore, an atom of the module *may not be an atom of the ontology*.

4.6 DIS Modularization

Up to this point we have introduced two modules: the view traversal module, and the principal ideal subalgebra module. Both of these modules are defined in terms of a domain ontology O. However, we now introduce how a DIS can be produced from the module's ontology.

Definition 9 (DIS Module). *Given a DIS $I = (O, \mathcal{A}, \tau)$, a DIS-module of I is the DIS $I_M = (O_M, \mathcal{A}_M, \tau_M)$ such that:*

1. *$O_M = (\mathcal{C}_M, \mathcal{L}_M, \mathcal{G}_M)$ is a module of O*
2. *$\mathcal{A}_M = (A_M, +, \cdot, -, 0, 1, c_k)_{k \in L_M}$*
3. *τ_M is the restriction of τ to only the elements of A_M.*

where L_M is the carrier set of the Boolean lattice $\mathcal{L}_M$, A is the carrier set of $\mathcal{A}$, and $A_M = \{a \mid a \in A \wedge \tau(a) \in L_M\}$.

When determining a DIS module, either a view traversal module or a principal ideal subalgebra module can be used as the O_M. The remaining components – the cylindric algebra and the τ function – are redefined from the original DIS for the DIS-module by using the new domain ontology's (i.e., the module's) Boolean lattice. With this definition of a DIS-module, a modularized DIS will result in a new DIS that can be further modularized or reasoned on.

5 Quantifying Knowledge Loss

The process of modularizing implies the loss of a part of the original ontology. The knowledge that could be obtained from the lost part becomes unattainable. By leveraging that the view traversal module is constructed using the principal ideal generated by

c_{st}, we are able to both determine the scope of the domain knowledge that is lost, and quantify it.

A Boolean lattice is the underlying structure of both the original ontology that is modularized, as well as the view traversal module. More specifically, the Boolean lattice of the view traversal module is a principal ideal of the Boolean lattice of the original ontology. Using this fact and the first theorem of isomorphism [58], there must be a homomorphism from the Boolean algebra associated to the original ontology to some other Boolean algebra to which the Boolean algebra associated to the view traversal module is its kernel. In other words, the existence of the view traversal module implies the existence of another Boolean algebra, and thus, a module that can be built around it. We denote the homomorphism which maps one Boolean algebra to another as $f : \mathcal{B}_1 \rightarrow \mathcal{B}_2$, where $\mathcal{B}_1$ is the Boolean algebra associated with the original ontology, and $\mathcal{B}_2$ is some other Boolean algebra. Let B_1 be the carrier set of $\mathcal{B}_1$, the kernel of f is the set of all elements from B_1 that are mapped to the identity element e_C of $\mathcal{B}_2$. The kernel of f contains only one element if and only if f is injective.

In the following, we construct an example of the function f. Let p_0 be any element from B_1. We define an instance of f as the f_{p_0} for a given $p_0 \in B_1$ and every $p \in B_1$:

$$f_{p_0}(p) = p \otimes p_0 \tag{4}$$

The kernel of f_{p_0} is defined as the set of all elements which map by f_{p_0} to the identity element:

$$ker(f_{p_0}) = \{p \in B_1 \mid f_{p_0}(p) = e_C\} \tag{5}$$

To illustrate the significance of the relationship between the view traversal module, the homomorphism, the kernel, and the knowledge lost, we return to the original Boolean lattice illustrated in Figure 2. We set $p_0 = \mathit{Taste}$, where $\mathit{Taste} = \mathit{Sugar} \oplus \mathit{Body}$. The use of Equation 4 signals the intent to map every concept from the Boolean lattice of the original ontology to a new Boolean lattice which is focused on the concept *Taste*. In this case, the function f_{p_0} in Equation 4 maps every concept from the original Boolean lattice to Taste or one of its parts. As the original Boolean lattice consists of 2^6 concepts, and the Boolean lattice with *Taste* as the top consists of 2^2 concepts, the mapping will not be injective. The elements that map to e_C will populate the kernel of f_{p_0}, as given in Equation 5.

For example, consider the following four evaluations where $p_0 = \mathit{Taste}$,

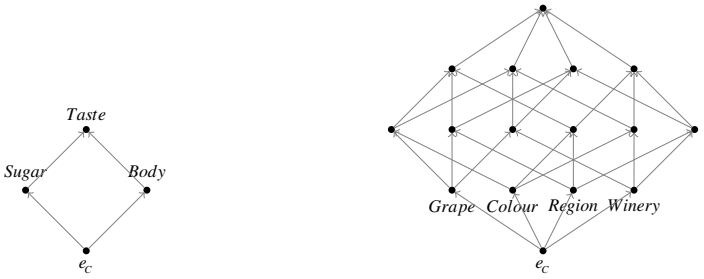
$$\begin{gathered} f_{p_0}(\mathit{Sugar}) = \mathit{Sugar} \otimes \mathit{Taste} = \mathit{Sugar}, \\ f_{p_0}(\mathit{Wine}) = \mathit{Wine} \otimes \mathit{Taste} = \mathit{Taste} \\ f_{p_0}(\mathit{Grape}) = \mathit{Grape} \otimes \mathit{Taste} = e_C \\ f_{p_0}(\mathit{Region} \oplus \mathit{Winery}) = (\mathit{Region} \oplus \mathit{Winery}) \otimes \mathit{Taste} = e_C \end{gathered}$$

The function f_{p_0} maps any value that is a part of *Taste*, such as *Sugar*, to itself, such that $f_{p_0}(\mathit{Sugar}) = \mathit{Sugar}$. Any value that an element of *Taste* is a part of, such as *Wine*, is also mapped to itself. Finally, any other element is mapped to the empty concept. Thus, with the starting concept $p_0 = \mathit{Taste}$, the kernel of the homomorphism f_{p_0} is composed of the concept $p_0' = \mathit{Grape} \oplus \mathit{Colour} \oplus \mathit{Region} \oplus \mathit{Winery}$ and all of its parts:

$$ker(f_{p_0}) = \{c \in L \mid c \sqsubseteq_C p_0'\} \tag{6}$$

Equations 4 and 5 together state that any concept $p \in B_1$ where $p \otimes p_0 = e_C$ is a part of $ker(f_{p_0})$. Using the definition of the concept complement and Boolean lattice meet, we get $p'_0 \otimes p = e_C$. Therefore, $p'_0 \in ker(f_{p_0})$. For any $c \sqsubseteq_C p'_0$, according to the definition of `partOf`, $c \sqsubseteq_C p'_0 \iff c \otimes p'_0 = c$. With this definition of `partOf` and Equation 4, we get $f_{p_0}(c) = (c \otimes p'_0) \otimes p_0$. Using the associativity of $\otimes$ operator, we get $f_{p_0}(c) = c \otimes (p'_0 \otimes p_0) = c \otimes e_C = e_C$. Therefore, any element $c \sqsubseteq_C p'_0$ also belongs to the kernel of f_{p_0}, which explains Equation 6.

Figure 9 displays both Boolean lattices: the one mapped to by f_{p_0} and the one constructed using $ker(f_{p_0})$.



The Boolean lattice mapped to by f_{p_0} The kernel of f_{p_0}

Fig. 9: $f_{p_0}(p) = p \otimes Taste$, where $p_0 = Taste$

The kernel of f_{p_0} for the example is populated by the concepts p'_0 and all its parts. This is the same set as the one obtained by determining the principal ideal generated by p'_0 on the original ontology. This aligns with the first isomorphism theorem which states the kernel is the carrier set for a principal ideal of the original Boolean algebra. In our example, p'_0 is the complement of *Taste* since it is the concept that is created from combining all atoms that are *not* part of *Taste*. Therefore we observe the following relationship between the three Boolean algebra structures. Let $\mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_3$ be three Boolean algebras, and p_0 be a concept from the carrier set B_1 of $\mathcal{B}_1$. Applying the function in Equation 4 with p_0 will result in a homomorphism that maps $\mathcal{B}_1$ to $\mathcal{B}_2$. This homomorphism has a kernel, which is itself associated with a Boolean algebra, $\mathcal{B}_3$. $\mathcal{B}_3$ corresponds to a module that is equivalent to, using the view traversal function defined in Definition 7, $vt_D(p'_0)$.

With this relationship between the Boolean lattices of the original ontology, the one determined by the homomorphism f_{p_0}, and its kernel, we revisit measuring the loss of knowledge. As introduced, the process of modularizing implies the possible loss of knowledge that can be derived from the omitted part of the ontology. The kernel of f_{p_0} represents a quantifiable measure of the lost part of the ontology. It is quantifiable because, as the kernel is a measure of how non-injective the homomorphism is: the larger a kernel is, the more concepts that were 'lost' in the mapping. This use of measuring the kernel to determine how much knowledge is lost aligns with our example in Fig 9. The chosen p_0 was one made of only two constituent concepts, *Sugar* and *Body*. It is

to be expected that, given the original ontology has six atoms and we are only using two, a significant amount of knowledge is lost. The kernel of f_{p_0} is a Boolean lattice formed from the four missing atoms, and represents all the concepts that are lost. Thus, from an ontologist's perspective, the manipulation of the homomorphism and the kernel allow for the control of what domain knowledge will be present in the module, or, what domain knowledge will *not* be there by ensuring it is in the kernel. In [35], an equation giving a measure of the knowledge loss is provided. It is based on comparing the number of atoms in the kernel to the total number of atoms in the original ontology. This can be made into a percentage which represents how many total concepts are lost. For our example, as we are using two of the six atoms, we are losing roughly 67% of the domain knowledge.

A view traversal modularization certainly results in the loss of knowledge which can be measured as described in the previous paragraph, but this does not capture all knowledge that is lost. For instance, the view traversal module does not preserve the complement operator, and the knowledge that is lost due to this is not captured using this approach. However, since the principal ideal subalgebra module preserves the complement operator, it does not lose this knowledge.

The atoms of a principal ideal subalgebra module might not be an atom in the ontology. It is possible that an atom of the principal ideal subalgebra module is a composite concept in the original ontology. This change in granularity of a concept from the original ontology to the principal ideal subalgebra module results in a more nuanced loss of knowledge. It is modifying how a concept is to be understood rather than simply removing a concept from the ontology, like view traversal does. Thus, the proposed approach for knowledge loss would not be sufficient for gathering all the knowledge lost in a principal ideal subalgebra module.

6 Discussion and Future Work

A major consequence of the view traversal module not being locally correct is that the complement operator is not preserved. As the lattice operations of join and meet are preserved, the knowledge that is deduced using these operators will be preserved, but anything deduced with the complement operator will not be. Contextually, this has its own significance: the view traversal module is a focused snapshot of the domain. It removes concepts from the context that is being evaluated by entirely removing the concepts and its parts. Given that a query for the complement of a concept is to ask for the concept that is built from the atoms of those that do not build the queried concept, it does not make sense for the answer to be a concept that is not a part of the context. Instead, the answer should be one which is a part of the context. However, there is still a significant need for being able to utilize the complement on a module, and be able to use the result without it being incorrect in the original context. For this, the principal ideal subalgebra module is useful. The cost of preserving the complement operator is the size of the principal ideal subalgebra module being greater than that of a view traversal module, as well as modifying the granularity of some concepts in the principal ideal subalgebra module. The application of either type of modularization approach can be seen dependent on priority of the engineer: view traversal produces lightweight

and easily usable modules, but do not preserve the complement operator, whereas the principal ideal subalgebra module is one that preserves all knowledge from the Boolean algebra operators, but are larger and contain composite concepts as atoms.

The use of the algebraic notion of a homomorphism associated to an ideal allows for the measurement of the knowledge that is lost due to modularization. This is a significant step towards being able to not only modularize based on preserving certain pieces of knowledge, but also by ensuring certain knowledge is not able to be determined via reasoning on the module. For fields related to privacy or determining the knowledge an autonomous agent can know via the module it has, this is a significant result. With the result of knowledge being quantified using the size of the kernel, we are able to determine the amount of domain knowledge that is being lost (or retained). However, as there are many different types, or facets of knowledge beyond simply the concepts themselves, there is work to be done in exploring how modularization on a DIS ontology can be characterized to capture these properties.

Lastly, the result of composite concepts being an atom in a principal ideal subalgebra module demonstrates insight to the different ways a single domain can be understood. A single non-decomposable concept to one agent might be a concept made of many smaller parts to another. In this case, the second agent has a *finer* understanding of that concept, whereas the first agent has a *coarser* understanding. A modularization technique that, instead of focusing on the omission or retention of concepts, but rather the coarsening or refinement of a concept could prove significant. By coarsening a concept, you produce a light-weight module at the cost of not being able to recognize or use the parts of the coarsened concept. In contrast, the refinement of a concept would result in a *larger* ontology, but distinguish the composing atoms of the refined concept.

7 Conclusion

In this paper, modularizing a DIS ontology with view traversal was introduced. The ability to leverage both the underlying Boolean lattice and the Boolean algebra associated with the ontology allows for a simple deterministic way to modularize the ontology. It is shown that the view traversal module is equal to a principal ideal, which is itself a Boolean sublattice. The Wine Ontology was utilized to demonstrate an example of extracting a view traversal module from a DIS ontology. Additional to the view traversal module, the principal ideal subalgebra module is introduced as a way of extending the view traversal module so that the complement operator is preserved.

Finally, the topic of measuring the knowledge that is lost due to modularization is introduced using the view traversal module. Using the first isomorphism theorem, we show the view traversal module is associated to the kernel of a homomorphism f that maps the Boolean algebra associated to the original ontology to another separate Boolean algebra. Since the kernel is a measure of the non-injectivity of a homomorphism, the size of the kernel is used to quantify the domain knowledge that is lost by the modularization process. It is also shown how the modularization process can be guided by ensuring the retention (or exclusion) of knowledge by their existence within the kernel. Other types of knowledge, such as the knowledge associated with the coarsening or refinement of a concept is introduced as a rich area for future work.

References

1. Algergawy, A., Babalou, S., Kargar, M.J., Davarpanah, S.H.: Seecont: A new seeding-based clustering approach for ontology matching. In: East European Conference on Advances in Databases and Information Systems. pp. 245–258. Springer (2015)
2. Algergawy, A., Babalou, S., König-Ries, B.: A new metric to evaluate ontology modularization. In: SumPre@ ESWC (2016)
3. Anand, N., van Duin, R., Tavasszy, L.: Ontology-based multi-agent system for urban freight transportation. International Journal of Urban Sciences **18**(2), 133–153 (2014)
4. 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.: Gene ontology: tool for the unification of biology. Nature genetics **25**(1), 25 (2000)
5. Baader, F.: The description logic handbook: Theory, implementation and applications. Cambridge university press (2003)
6. Baader, F., Borchmann, D., Nuradiansyah, A.: The identity problem in description logic ontologies and its application to view-based information hiding. In: Joint International Semantic Technology Conference. pp. 102–117. Springer (2017)
7. Babalou, S., Kargar, M.J., Davarpanah, S.H.: Large-scale ontology matching: A review of the literature. In: Web Research (ICWR), 2016 Second International Conference on. pp. 158–165. IEEE (2016)
8. Belmonte, M.V., Pérez-de-la Cruz, J.L., Triguero, F.: Ontologies and agents for a bus fleet management system. Expert Systems with Applications **34**(2), 1351–1365 (2008)
9. Benomrane, S., Sellami, Z., Ayed, M.B.: An ontologist feedback driven ontology evolution with an adaptive multi-agent system. Advanced Engineering Informatics **30**(3), 337–353 (2016)
10. Bezerra, C., Freitas, F., Euzenat, J., Zimmermann, A.: Modonto: A tool for modularizing ontologies. In: Proc. 3rd workshop on ontologies and their applications (Wonto). pp. No–pagination (2008)
11. Bienvenu, M., Kikot, S., Kontchakov, R., Podolskii, V.V., Zakharyaschev, M.: Ontology-mediated queries: Combined complexity and succinctness of rewritings via circuit complexity. Journal of the ACM (JACM) **65**(5), 28 (2018)
12. Bodenreider, O.: The unified medical language system (umls): integrating biomedical terminology. Nucleic acids research **32**(suppl_1), D267–D270 (2004)
13. Brickley, D., Miller, L.: Foaf vocabulary specification 0.91 (2010)
14. Codd, E.F.: A relational model of data for large shared data banks. vol. 26, pp. 64–69. ACM New York, NY, USA (1983)
15. Cuenca Grau, B.: Privacy in ontology-based information systems: A pending matter. Semantic Web **1**(1, 2), 137–141 (2010)
16. Davey, B.A., Priestley, H.A.: Introduction to lattices and order. Cambridge university press (2002)
17. De Giacomo, G., Lembo, D., Lenzerini, M., Poggi, A., Rosati, R.: Using ontologies for semantic data integration. In: A Comprehensive Guide Through the Italian Database Research Over the Last 25 Years, pp. 187–202. Springer (2018)
18. Del Vescovo, C., Parsia, B., Sattler, U., Schneider, T.: The modular structure of an ontology: Atomic decomposition and module count. In: WoMO. pp. 25–39 (2011)
19. Dietz, J.L.: What is Enterprise Ontology? Springer (2006)
20. Dijkstra, E., Scholten, C.: Predicate Calculus and Program Semantics. Springer-Verlag New York, Inc., New York, NY, USA (1990)
21. Domingue, J., Dzbor, M.: Magpie: supporting browsing and navigation on the semantic web. In: Proceedings of the 9th international conference on Intelligent user interfaces. pp. 191–197. ACM (2004)

22. Freitas, A., Panisson, A.R., Hilgert, L., Meneguzzi, F., Vieira, R., Bordini, R.H.: Applying ontologies to the development and execution of multi-agent systems. In: Web Intelligence. vol. 15, pp. 291–302. IOS Press (2017)
23. Grau, B.C., Parsia, B., Sirin, E.: Combining owl ontologies using e-connections. Web Semantics: Science, Services and Agents on the World Wide Web **4**(1), 40–59 (2006)
24. Grau, B.C., Parsia, B., Sirin, E., Kalyanpur, A.: Modularity and web ontologies. In: KR. pp. 198–209 (2006)
25. Gries, D., Schenider, F.: A Logical Approach to Discrete Math. Springer Texts And Monographs In Computer Science, Springer-Verlag, New York (1993)
26. Halmos, P.R.: Lectures on Boolean algebras. Courier Dover Publications (2018)
27. Harding, J., Heunen, C., Lindenhovius, B., Navara, M.: Boolean subalgebras of orthoalgebras. Order pp. 1–47 (2017)
28. HENKIN, L., MONK, J., TARSKI, A.: Cylindric algebras. ii. Studies in logic and the foundations of mathematics **115** (1985)
29. Hirsch, R., Hodkinson, I.: Relation algebras by games. Elsevier (2002)
30. Hustadt, U., Motik, B., Sattler, U.: Data complexity of reasoning in very expressive description logics. In: IJCAI. vol. 5, pp. 466–471 (2005)
31. Jaskolka, J., MacCaull, W., Khedri, R.: Towards an ontology design architecture. In: Proceedings of the 2015 International Conference on Computational Science and Computational Intelligence. pp. 132–135. CSCI 2015 (2015)
32. Jiménez-Ruiz, E., Kharlamov, E., Zheleznyakov, D., Horrocks, I., Pinkel, C., Skjæveland, M.G., Thorstensen, E., Mora, J.: Bootox: Practical mapping of rdbs to owl 2. In: International Semantic Web Conference. pp. 113–132. Springer (2015)
33. Kantamneni, A., Brown, L.E., Parker, G., Weaver, W.W.: Survey of multi-agent systems for microgrid control. Engineering applications of artificial intelligence **45**, 192–203 (2015)
34. Khan, Z.C., Keet, C.M.: Toward a framework for ontology modularity. In: Proceedings of the 2015 Annual Research Conference on South African Institute of Computer Scientists and Information Technologists. p. 24. ACM (2015)
35. LeClair., A., Khedri., R., Marinache., A.: Toward measuring knowledge loss due to ontology modularization. In: Proceedings of the 11th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management - Volume 2: KEOD. INSTICC, SciTePress (2019)
36. Legat, C., Seitz, C., Lamparter, S., Feldmann, S.: Semantics to the shop floor: towards ontology modularization and reuse in the automation domain. IFAC Proceedings Volumes **47**(3), 3444–3449 (2014)
37. Lutz, C., Walther, D., Wolter, F.: Conservative extensions in expressive description logics. In: IJCAI. vol. 7, pp. 453–458 (2007)
38. Maedche, A., Motik, B., Stojanovic, L., Studer, R., Volz, R.: Ontologies for enterprise knowledge management. IEEE Intelligent Systems **18**(2), 26–33 (2003)
39. Marinache, A.: On the Structural Link Between Ontologies and Organised Data Sets. Master's thesis (2016)
40. Motik, B.: Reasoning in description logics using resolution and deductive databases. Ph.D. thesis, Karlsruhe Institute of Technology, Germany (2006)
41. Movaghati, M.A., Barforoush, A.A.: Modular-based measuring semantic quality of ontology. In: Computer and Knowledge Engineering (ICCKE), 2016 6th International Conference on. pp. 13–18. IEEE (2016)
42. Nadal, S., Romero, O., Abelló, A., Vassiliadis, P., Vansummeren, S.: An integration-oriented ontology to govern evolution in big data ecosystems. Information Systems **79**, 3–19 (2019)
43. Noy, N., Musen, M.: Traversing ontologies to extract views. Modular Ontologies pp. 245–260 (2009)

44. Noy, N.F., McGuinness, D.L., et al.: Ontology development 101: A guide to creating your first ontology (2001)
45. Noy, N.F., Musen, M.A.: Specifying ontology views by traversal. In: International Semantic Web Conference. vol. 3298, pp. 713–725. Springer (2004)
46. Pakdeetrakulwong, U., Wongthongtham, P., Siricharoen, W.V., Khan, N.: An ontology-based multi-agent system for active software engineering ontology. Mobile Networks and Applications **21**(1), 65–88 (2016)
47. Poggi, A., Lembo, D., Calvanese, D., De Giacomo, G., Lenzerini, M., Rosati, R.: Linking data to ontologies. In: Journal on data semantics X, pp. 133–173. Springer (2008)
48. Raimond, Y., Abdallah, S.A., Sandler, M.B., Giasson, F.: The music ontology. In: ISMIR. vol. 2007, p. 8th. Citeseer (2007)
49. Rector, A., Rogers, J., Pole, P.: The galen high level ontology (1996)
50. Romero, A.A., Kaminski, M., Grau, B.C., Horrocks, I.: Module extraction in expressive ontology languages via datalog reasoning. Journal of Artificial Intelligence Research **55**, 499–564 (2016)
51. Santos, E., Faria, D., Pesquita, C., Couto, F.M.: Ontology alignment repair through modularization and confidence-based heuristics. PloS one **10**(12), e0144807 (2015)
52. Sikorski, R.: Boolean algebras, vol. 2. Springer (1969)
53. Spackman, K.A., Campbell, K.E., Côté, R.A.: Snomed rt: a reference terminology for health care. In: Proceedings of the AMIA annual fall symposium. p. 640. American Medical Informatics Association (1997)
54. Stone, M.H.: The theory of representation for boolean algebras. Transactions of the American Mathematical Society **40**(1), 37–111 (1936)
55. Stuckenschmidt, H., Klein, M.: Structure-based partitioning of large concept hierarchies. In: International semantic web conference. vol. 3298, pp. 289–303. Springer (2004)
56. Suchanek, F.M., Kasneci, G., Weikum, G.: Yago: A large ontology from wikipedia and wordnet. Web Semantics: Science, Services and Agents on the World Wide Web **6**(3), 203–217 (2008)
57. Wache, H., Voegele, T., Visser, U., Stuckenschmidt, H., Schuster, G., Neumann, H., Hübner, S.: Ontology-based integration of information-a survey of existing approaches. In: IJCAI-01 workshop: ontologies and information sharing. vol. 2001, pp. 108–117. Seattle, USA (2001)
58. Van der Waerden, B.L., Artin, E., Noether, E.: Moderne algebra, vol. 31950. Springer (1950)
59. Xiao, G., Calvanese, D., Kontchakov, R., Lembo, D., Poggi, A., Rosati, R., Zakharyaschev, M.: Ontology-based data access: A survey. IJCAI (2018)
60. Xu, D., Karray, M.H., Archimède, B.: A knowledge base with modularized ontologies for eco-labeling: Application for laundry detergents. Computers in Industry **98**, 118–133 (2018)
61. Xue, X., Tang, Z.: An evolutionary algorithm based ontology matching system. Journal of Information Hiding and Multimedia Signal Processing **8**(3), 551 – 556 (2017), high heterogeneity;Matcher combination;Matching system;Ontology matching;Optimal model;Recall and precision;Semantic correspondence;State of the art;
62. Zhou, L., Pan, M., Sikorski, J.J., Garud, S., Aditya, L.K., Kleinelanghorst, M.J., Karimi, I.A., Kraft, M.: Towards an ontological infrastructure for chemical process simulation and optimization in the context of eco-industrial parks. Applied Energy **204**, 1284–1298 (2017)
63. Ziegler, P., Dittrich, K.R.: Three decades of data intecration—all problems solved? In: Building the Information Society, pp. 3–12. Springer (2004)