Attribute Lattices: A Schema-free Unified Framework for Data Semantics

Mojtaba Asgari[1], Jeffrey Parsons[1], and Yair Wand[2]

[1]Faculty of Business Administration, Memorial University of Newfoundland, St. John's, NL, CANADA
{m.asgari, jeffreyp}@mun.ca
[2]Sauder School of Business, The University of British Columbia, Vancouver, BC, CANADA
yair.wand@ubc.ca

Abstract. One key characteristic of big data is variety. With massive and growing amounts of data existing in independent and heterogeneous (structured and unstructured) sources, assigning consistent data semantics, which is essential for making sense of data sources, is an increasingly important challenge. We use ontology and human cognitive principles (i.e., classification theory) to formally define the concept of ***attribute lattice***. An attribute lattice is a *graph-based, schema-free conceptual model that represents attributes of instances in the domain of interest and precedence relations among them.* The class structure of the domain can be inferred from the precedence relations in the lattice. In other words, in an attribute lattice, both properties and classes are represented as attributes – they are distinguished only by the pattern of arcs and nodes that surround them (the ***semantic neighbourhood***). We propose that this form of representation offers a unified framework for modeling data that can be used to resolve semantic data heterogeneity.

Keywords: Attribute lattice, Instance-Based Data Model (IBDM), Semantic data integration, Property precedence.

1 Introduction

Semantic data heterogeneity (a form of variety) is an active research area in several research communities such as databases, domain ontologies and big data [1-3]. In spite of its pervasiveness and the substantial work in this area, resolving semantic heterogeneity remains a key challenge in using data from multiple independent sources. The lack of deep data understanding, and a focus on syntax and structure, rather than on data semantics, hinders semantic data integration [4, 5].

Two common assumptions in database design contribute to the challenge of understanding the semantics of data. First, traditional database design assumes (either explicitly or implicitly) that *instances must belong to a class to exist in a database* [6]. Based on this assumption, the main body of research on resolving semantic heterogeneity focuses on schema mapping techniques [1-3]. Second, as a result of being schema-oriented, database design approaches assume *there is a clear and fundamental* *distinction between classes and properties of instances* (i.e., instances belong to the classes and possess properties).

In this paper, we propose a schema-free (i.e., liberated from fixed schemas) conceptual modeling grammar that can be used to resolve semantic heterogeneity. Our approach is based on the premise that the semantics of data is a function of human cognition. Therefore, this approach uses cognitively-based instance-level constructs (attributes of instances and relations among them) to represent classes and properties in a lattice structure. We argue that the notion of property or class is contextual, such that a specific attribute (node in a lattice) can designate either a property of an instance or a class, depending on the context (the immediately connected section of the lattice).

Our approach is based on the instance-based data model (IBDM) [6]. In conformance with principles from cognitive psychology and philosophical ontology [7], the IBDM argues that instances (things) exist independent of classes, and classes are derived constructs that provide useful abstractions [6]. The IBDM proposes a two-layered structure in which one layer is responsible for the storage of data about individual entities (instances) and their attributes, and the other keeps track of the definition of classes in terms of attributes of instances. In the IBDM approach, instances are stored only with their attributes, rather than classes [6]. By freeing data from predefined classes, this approach simplifies the semantic integration of data by eliminating the need to map class-level constructs between heterogeneous data sources.

We use the concept of ***property precedence*** [8-11] to propose a graph-based structure that we call an ***attribute lattice***. The attribute lattice provides a formalism to express subsumption relationships between attributes. If **r** and **s** are two attributes, **s** *precedes* **r** (denoted as **r**→**s**) if and only if any instance possessing **r** also possesses **s**. For instance, if **r** is "ability to walk" and **s** is "ability to move," every instance that possesses "ability to walk," also possesses "ability to move" ("ability to walk"→"ability to move").

In an attribute lattice, nodes represent attributes, and arcs show precedence relations among attributes. The attributes represent concepts. Depending on its pattern of inbound and outbound arcs, an attribute can designate a property, a category, or a class[1]. The key distinguishing feature of the attribute lattice (compared to other graph-based data models) is that the difference between the type of concept (property, category, or class) is purely contextual. The type of attributes in the lattice are interpreted solely based on the structure of precedence relations, reflecting a human view of links among attributes.

In the following, we begin by discussing related literature (Section 2). Then we formally define the proposed attribute lattice and examine some of its properties (Section 3). Finally, we summarize our research contribution and discuss opportunities for further research (Section 4).

2 Related Research

In this section, we discuss principles from cognitive psychology and philosophical ontology that guide us in defining an attribute lattice conceptual modeling grammar.

[1] Hereafter in this paper, attribute refers to the node itself, and property denotes one type of node.

Then, we briefly review approaches for resolving semantic heterogeneity through data integration to highlight a common assumption underlying many approaches – the reliance on class-based schemas – and to point out that this dependency in turn leads to several known challenges in these approaches. (For comprehensive reviewer of semantic integration approaches, see, for example, [12, 13].)

2.1 Principles from cognitive psychology and philosophical ontology

In this paper, we use principles from philosophical ontology and cognitive psychology. In particular, we use Bunge's ontology [8], as elaborated for conceptual modeling by Wand and Weber [7], [14], which is widely known and used. In particular, three ontological principles are central to our approach: (1) the world consists of substantial things that are assumed to exist physically; (2) things possess attributes; and (3) subsumption relations between attributes can be expressed by property precedence [7, 9, 14]. The concept of property precedence has been used for improving the semantics of conceptual models [11].

Principles of human cognition provide additional grounding in developing the concept of attribute lattice. Lakoff [15] argues that humans understand the world by classifying things. In contrast to the common assumption in schema-based and domain ontology-based approaches, classes do not exist independent of human cognition. Indeed, classes are useful abstractions that support inferences about instances based on partially observed information [16, 17].

2.2 From Traditional to Domain Ontology-based Data Integration

Semantic data integration is an approach for providing integrated access to disparate and semantically heterogeneous data [18]. The field has been an active area of research since the 1980s [19, 20]. However, in spite of abundant literature, concerns about the lack of "consistent theory and methodology", and "in-depth understanding of semantics" have persisted [5].

Traditional semantic data integration can be divided into two main steps [2, 21]. The first step, a match operation, takes two schemas as input and provides a semantic mapping between schema elements. The second step proceeds to define mapping expressions formally. Depending on the context, the mapping can be expressed in different languages such as SQL, LAV (local as view), or GAV (global as view). In these methods, the data reside in data sources, while the global schema provides a unified, integrated, and virtual view [22].

Generally speaking, matcher types can be categorized into schema level and data (instance) level matchers [2]. As argued by [6], in traditional data models, classification is inherently part of data management and storage. In this regard, schema reconciliation is a prerequisite to accessing data. Not surprisingly, then, the main body of semantic data integration literature focuses on schema integration and data integration based on a so-called global schema (or a mediated schema). Data level matchers are often used as a complementary method or for semi-structured data when a schema cannot be constructed from data. These methods are either based on linguistic characteristics (for text elements) such as keywords relative frequency and string match (e.g. [23]), or constraint characteristics (for more structured data), such as value ranges and averages [2]. Probabilistic and statistical models are the key common approaches used in data level matchers (e.g. [24, 25]).

The initial approach for data integration was hard-coding the integration points. In this approach, developers were supposed to implement separate and specific code to get access to components of other schemas. Therefore, it had no flexibility, and it was hard to maintain. Although subsequent methods were loosely coupled and easier to manage, data semantics was a missing component in the integration process [4]. Domain ontology-based approaches were introduced to address this lack of semantics. A domain ontology has two primary roles to play in these methods [26]: first, creating a mapping between concepts with fixed classes (ontology) and the content; and second, integrating these concepts from different ontologies. Although schemas and ontologies have differences (for a detailed comparison see [4]), since both using fixed classes, similar techniques were used for schema mapping and ontology mapping [12]. The ontology mapping techniques, like their ancestors (schema mapping), suffer from a lack of deep (cognitive) semantics and their ties to schema and fixed classes.

2.3 Semantic Web and Linked-Data

The notion of Semantic Web, first coined by Tim Berners-Lee [27], has been introduced to semantically integrating semi-structured data on the web. To achieve this goal, Linked Data provides a set of best practices (new paradigm), and offers principles [28, 29] to publish and interlink machine-readable data on the web [29]. In brief, Linked Data uses URIs [30] to define uniquely identifiable web resources and RDF [31] triples (subject, predicate, and object) to encode how these resources are related [32].

As a semantic extension of the RDF data model, RDF schema [33] provides a data model vocabulary (schema) for RDF-based data sets. It provides mechanisms to describe groups of resources in terms of classes and properties by using RDF-based syntax [33]. During the past two decades, multiple web ontologies such as OIL [34], DAML + OIL[35], OWL [36], and OWL2 [37] have been introduced to represent information about the structure of these resources. The primary goal of these ontologies is to provide structured data that can be used for inference and reasoning. Although some of these ontologies improved the capability of RDF schema in important ways such as adding subsumption hierarchy to the classes and properties [38], the clear distinction between class and property is a key assumption in all schemas.

2.4 Known Issues in Schema-Based Approaches

There are several well-known problems in schema-based approaches that potentially can be addressed by the proposed model: (1) merging one or more properties in one schema with a class in another (e.g. [39-41]; (2) matching more general concept (class or property) to more specific concept (e.g. [42-44]); and (3) complex matching in which possessing several concepts (class or property) at the same time in one schema is semantically equal to possessing one concept (or several concepts) in the second schema (e.g. [42, 43] .)

In the following, first, the attribute lattice is defined, and then, the possible ways that this model can tackle to above mentioned longstanding issues are discussed.

3 Attribute Lattice Definitions and Components

This study has two key premises. First, classification guidelines based on human cognition [17] contribute to gaining a deeper understanding of data. Second, a barrier to semantic integration is the dependency on schemas or fixed classifications [6, 9].

Classification is a mechanism to identify instances and infer further information about them [45]. We argue that classes and properties express statements about instances and whether a particular statement is considered a class or a property is based on the relationship between a statement (attribute) and other attributes linked to it. In other words, the first premise states that an attribute (node) in the lattice expresses a statement about instances; the type of attribute expressed in the statement (class, category, property) depends on the pattern of arcs and nodes linked to it. Thus, the type of any attribute is determined by the pattern of attributes connected to it via arcs, and may change over time.

Definition 1 (Semantic neighbourhood): The ***semantic neighbourhood*** of an attribute A is the set of nodes and arcs reachable from A where an arc is either directly connected to A, or connected via the attributes in its ***full expansion*** [17].

As discussed later, in an attribute lattice, the union of all attributes in the base(s) of a class and derived attributes constitutes the full expansion of a class attribute, and the union of all subcategories of a category attribute constitutes its full expansion. In other words, the first definition asserts that nodes and arcs which are connected to a class/category via its full expansion are considered as the semantic neighbourhood of the A.

Principles from philosophical ontology suggest that ***property precedence*** can provide semantics for attributes. Property precedence provides a formalism to represent subsumption relations between attributes. Assume **r**,**s**∈P are two attributes; s precedes r (denoted as **r→s**) if and only if any instance that possesses **r** also possesses **s** (see [17] for a detailed discussion.)

Attributes in a lattice can be manifestations of higher-level attributes [9], and, such higher-level attributes can also support semantic integration of lattices. In particular, the relation between specialized and general attributes is a precedence relationship [9]. For example, *'is a student' (Stu)* precedes *'is a graduate student' (Grd)*, meaning that anyone who is a graduate student is a student.

Eq. (a) $\{\text{Grd}\}\rightarrow\{\text{Stu}\}$

In the following, the components of the attribute lattice are formally defined. In addition, examples from a university context are used to illustrate the lattice concepts.

3.1 Attribute Lattice Component Definition

Definition 2 (domain of interest): A ***domain of interest*** is a set of phenomena (instances), X, and a set of attributes, P, possessed by the instances in X (each attribute in P is possessed by at least one instance in X) [9, 17].

Also, we use $f(x_i)=\{P_1,P_2,\ldots,P_j\}$ to denote a function that returns all attributes possessed by specific instance x_i. In this definition, ***attribute*** (***property*** - using the notation in [17]) refers to any true statement (predicate) describing instances.

An attribute lattice has two main components. Nodes (circles) represent ***attributes*** (true statement describing instances). Directed arcs (arrows) represent precedence relations between attributes.

Definition 3 (Multiple property precedence): Assume R and S are non-empty sets of attributes such that $R,S\subseteq P$, and $R\cup S\neq R$ and $R\cup S\neq S$. A ***(multiple) property precedence (MP)*** exists between R and S (denoted by $R\rightarrow S$) if and only if every instance that possesses all attributes in R (for brevity, 'possesses R') also possesses all attributes in S [9, 17].

Definition 4 (Category): Assume R is a non-empty set of attributes. R is a ***category*** if and only if there exists at least one instance that possesses all attributes in R[9, 17]. A category is also referred to as a potential class using the terminology in [17].

For example, in a university context, assume that an instructor is either a faculty member or a graduate student. Three attributes are expressed here *'is a faculty member' (Fac)*, *'is an instructor' (Ins)*, and *'is a graduate student' (Grd)*.[2] Since instances that possess *{Ins}*, also possess either *{Fac}* or *{Grd}*, it is reasonable to have categories to have shorthand access to these sets of attributes. Two possible categories are *'is a graduate instructor' (GrdIns)* and *'is a faculty instructor' (FacIns)*.

Definition 5 (Subcategory precedence): Assume R and S are two sets of attributes such that $S\subset R\subseteq P$. Since $S\subset R$, by definition, any instance that possesses R possesses S. A ***subcategory precedence*** exists between R and S if and only if $S\subset R$ and at least one instance exists that possesses R and one instance that possesses S but not R. This particular type of precedence is denoted by an arc labeled with S ($R \xrightarrow{S} S$).

To illustrate, consider the example mentioned above. The following subcategory relations exist:

Eq. (b) $\{FacIns\} \xrightarrow{S} \{Fac\}$; and $\{FacIns\} \xrightarrow{S} \{Ins\}$
$\{GrdIns\} \xrightarrow{S} \{Grd\}$; and $\{GrdIns\} \xrightarrow{S} \{Ins\}$

Based on cognitive economy and inference, [16] offers criteria for evaluating these possible categories and choosing among them. A category (potential class) is a ***useful class*** whenever it has a ***base*** - a strict subset of attributes that is sufficient to identify an instance as a member of the class, and from which the remaining attributes of the class can be inferred [17]. In other words, to be useful, a class must provide information (in term of new attributes) about its members beyond the attributes required to identify

[2] Note that these labels might be considered "classes" in many contexts. For our purposes (and as discussed later), we do not distinguish between properties and classes at the node level (all nodes are attributes), but only distinguish a node as a class based on patterns in its semantic neighbourhood.

members as belonging to the class. In the attribute lattice, an attribute will be considered as a class only if it is a useful class with this definition.

Definition 6 (Base precedence): Assume R and S are two sets of attributes such that S⊂R⊆P. If S → R (R precedes S), R is a class and, S is a base for R. This is called ***base precedence*** and denoted by $R\xrightarrow{B}S$.

For instance, assume all (and only) instructors (either graduate or faculty) have a separate contract for their teaching and get a course-based salary for the course. In this situation, *'is an instructor'* is a class; and *'has an instructor's contract' (InsCnt)* is a base for this class. Also, *'has a course-based salary' (CbSlry)* and *'has a course to teach' (Crs)* are two derived attributes for this class.

Eq. (c) *{Ins}* is a class; *{InsCnt}*$\xrightarrow{B}$*{Ins}*; *{InsCnt}→{CbSlry, Crs};*

Simple and multiple precedence are semantically equivalent, so simple arcs represent them in the lattice (Fig 1.a). Subcategory precedence is shown by an arc labeled with S (Fig 1.b). Base precedence is represented by an arc labeled with B (Fig 1.c).

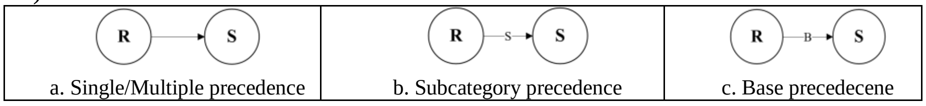


Fig. 1. Types of precedence

3.2 Attribute Lattice Characteristics

Lattice characteristics and rules contribute to reasoning about attributes and creating a valid attribute lattice. Consider that, in the attribute lattice, the type of an attribute depends on its semantic neighbourhood. The first set (of lattice characteristics) specifies how to infer the type of any node. The second set (of lattice rules) ensures that the lattice is valid and eliminates redundancies.

Examples based on the lattice representations of equations (a) to (c) above are shown in Fig 2.

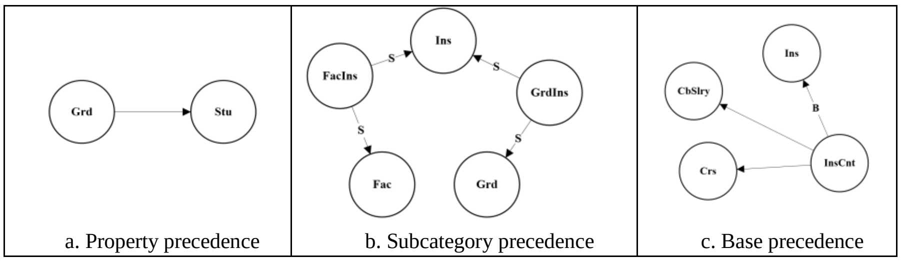


Fig. 2. Lattice representation of the precedences

Type of Attributes.

Characteristic 1 (Node type): An attribute (node) S in a lattice can be a property, category or class, based on the semantic neighbourhood of the node (its incoming and outgoing precedences). Each node in a lattice has only one type at a time.

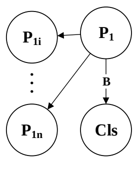
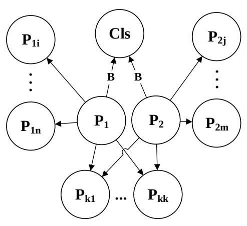
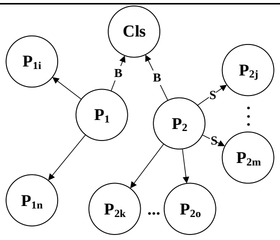
Characteristic 2 (Class): A node S in the lattice represents a class if and only if it has incoming base precedence. In other words, S is a class if and only if it precedes R such that R is a base for S.

For instance, in Fig.2(c), *{Ins}* precedes *{InsCnt}* with base precedence relation between *{Ins}* and *{InsCnt}*; therefore, {Ins} is a class. In this example, the class has only one base, and a property (a single attribute) is a base for it. However, it is possible for a class to have several attributes (a category) as a base and have more than one base (Fig.3). The union of base attributes and derived attributes defines the full expansion of a class [17]. When a class has more than one base, all bases form part of the class definition. Figure 3.b and 3.c show classes with more than one base. Note that, although for representation simplicity bases and derived attributes are shown as independent sets in Fig.3.c, these sets may have shared attributes.

Characteristic 3 (Category): A node R in the lattice denotes a category if and only if it has outgoing subcategory precedence.

A category in a lattice is a set of attributes (possessed by at least one instance). These attributes (sub-categories) can be property, class or even category. For instance, Fig.2.b shows two simple categories. In this figure, *{FacIns}* is a category with two subcategories, *{Fac}* and *{Ins}*. In other words, this part of the lattice states that any instance that *'is a faculty instructor,' 'is a faculty member'* **and** *'is an instructor.'*

A category in the lattice can, at the same time, be a base for a class. For instance, in Fig.3.c the class *{Cls}* has two bases – one of them *{P_2}* is a category.


Base: {P_1};
Derived: P_{1i},…,P_{1n}}
 	
{P_1,P1$_i$,…,P_{1n}}⊆P
	 P_{1i}
	Cls
	P_{2j}
	B
	B
	P_{1n}
	P_1
	P_2
	P_{2m}
	P_{k1}
	...
	P_{kk}
	Base: {P_1};
	Derived: {P_{1i},…,P_{1n},P_{k1},…,P_{kk}}
	
	Base: {P_2};
	Derived: {P_{2j},…,P_{2m},P_{k1},…,P_{kk}}
	{P_1,P_{1i} ,…,P_{1n},P_2,P_{2i},…,P_{2n},P_{k1},…, P_{kk}}⊆P
		 Cls
		P_{1i}
		B
		B
		P_{2j}
		S
		P_1
		P_2
		S
		P_{2m}
		P_{1n}
		P_{2k}
		...
		P_{2o}
		Base: {P_1};
		Derived: {P_{1i},…,P_{1n}}
		
		Base: {P_{2j},…,P_{2m}};
		Derived: {P_{2k},…,P_{2o}}
		{P_1,P_{1i},…,P_{1n},P_2,P_{2j},…,P_{2m},P_{2k},…,P_{2o}}⊆P

a. Class with one base	b. Class with two bases	c. Class with two bases; one property and one category

Fig. 3. Class with one and more than one base

Characteristic 4 (Property): An attribute R in the lattice represents a property if and only if it is neither a category nor a class.

Based on the first characteristic, attributes are class/category or property based on their semantic neighbourhood. Also, attributes can have one type at a time. Classes and categories are easily inferable based on the 2nd and 3rd characteristics and by considering their semantic neighbourhood (incoming and outgoing arcs). Consequently, the remaining attributes are properties.

Validation and Redundancy Rules.

Rule 1 (Class validation): Assume S precedes R with the base precedence relation (R can be a property or a category). S is a valid class if and only if, it has at least one attribute T that precedes R.

This rule ensures that the class provides information (in term of attributes) for class members *beyond what is needed to identify the members* (base attributes). For instance, consider Fig.4.a, in which R is preceded by S with base precedence. However, S is not a valid class. Attributes *{R_1, …, R_m}* are used to identify the class members, but no further attribute is inferable from class membership. In contrast, Fig.4.b shows a valid class.

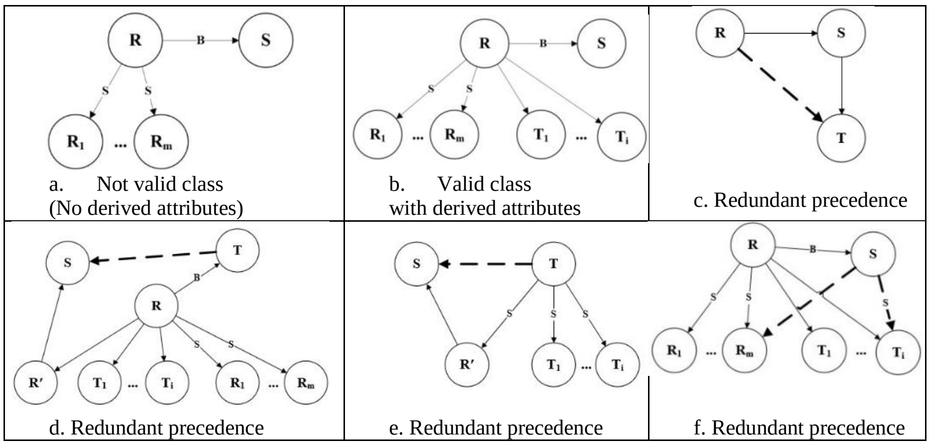


Fig. 4. Redundant Precedences

Rule 2 (Redundancy Rule): One desirable quality (for simplicity) of an attribute lattice is to represent only ***non-redundant*** relationships. The precedence relation (an arc) in the lattice is considered redundant if it can be inferred from (i.e., is implied by) other precedence relations. The following guidelines provide mechanisms to eliminate unnecessary precedences in the lattice.

First, the precedence between two nodes is redundant if the precedence relationship can be inferred from a transitive chain of other precedences. To illustrate, assume that $R \rightarrow S$ and $S \rightarrow T$. The precedence definition, $R \rightarrow T$ is thereby implied (Fig.4.c), so showing this precedence in an attribute lattice is redundant. In general, we propose to construct attribute lattices such that they retain only the longest chain of precedences between any two nodes.

Second, a precedence relation in the lattice is considered non-redundant if no proper subset of preceded attributes is sufficient to infer the preceding attribute [17]. Given $R \rightarrow S$, R is ***non-redundant*** if no $R' \subset R$ exists, such that $R' \rightarrow S$. To illustrate, assume that T is a class/category, R′ is a subset of this class/category ($R' \subset T$), and $R' \rightarrow S$. Although S precedes T ($T \rightarrow S$), this precedence is redundant (Fig.4.d and Fig.4.e).

Third, a subcategory precedence or multiple precedence from a class to its bases and all derived attributes is redundant. In other words, depicting direct precedences (either subcategory or multiple precedence) from the class to the attributes that belong to its full expansion is redundant (Fig.4.f).

3.3 Example

To demonstrate the attribute lattice concept, consider the following example in the context of a university (Fig.5). Assume, in this domain*, ‘is a student’ (Stu)* is an attribute. Also, *‘is registered’ (Reg)* is a sufficient condition to be a student, so that *{Reg}* is a base for *{Stu}* and *{Stu}* is a class. Moreover, a student *‘has a degree program’ (Deg)* and *‘has a start date’ (StrtDt)*; these two attributes are shown in Fig.5 as derived attributes for this class.

{Stu}={Reg, Deg, StrtDt} is a class;

Base attributes: {Reg}; Derived attributes: {Deg, StrtDt};

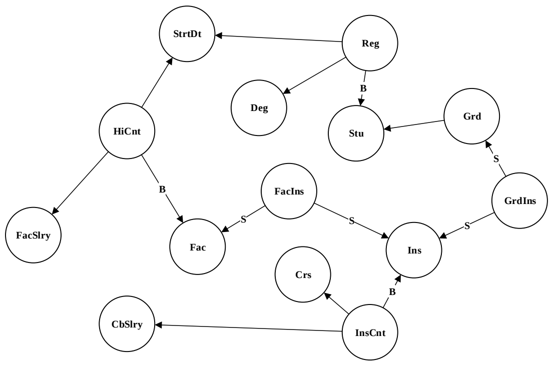


Fig. 5. Attribute lattice

Likewise, assume *‘has a hiring contract’ (HiCnt)* is a base for *‘is a faculty member’ (Fac)*. Also, assume faculty member *‘has a faculty base salary’ (FacSlry)* and *‘has a start date’ (StrtDt)* for her/his contract.

{Fac} = {HiCnt, FacSlry, StrtDt} is a class;

Base attributes: {HiCnt}; Derived attributes {FacSlry, StrtDt};

Furthermore, assume instructors have a separate contract for their teaching. Correspondingly, *‘is an instructor’ (Ins)* is another class in the attribute lattice. This attribute is a class because it has *‘has an instructors contract’ (InsCnt)* as a base. This class, as shown in the Fig.5, has two derived attributes which are; *‘has a course based salary’ (CbSlry)*, and *‘has a course to teach’ (Crs).*

{Ins}={InsCnt, CbSlry, Crs} is a class;

Base attributes: {InsCnt}; Derived attributes:{CbSlry, Crs};

Moreover, assume that both graduate students and faculty members can be instructors. *‘Is a graduate student’ (Grd)* is a specialized attribute of the existing one *{Stu}*. In this situation, it is reasonable to have another attribute *{Grd}* preceded by *{Stu}*. Since instances possess either *{Fac}* and *{Ins}* or *{Grd}* and *{Ins}* at the same time, two categories (FacIns, GrdIns) are defined in the lattice (Fig.5) to have the shorthand access to these sets.

{FacIns} = {Fac, Ins} *{GrdIns} = {Grd, Ins}*

Table1 provides the full list of all attributes in the lattice (Fig.5) in addition to their types and their full expansions.

The type of attributes represented in Table 1 considers the semantic neighborhood of nodes. However, based on the lattice fundamental assumptions, these types may change by integrating lattices or by additional domain-related information. Furthermore, higher-level attributes can be added to the lattice for lattice integration.

Table 1. The list of attributes in the Fig.5 and their types

Attribute description	*Label*	*Type*	*Full Expansion*
is a student	Stu	Class	{Reg, Deg, StrtDt}
is registered as a student	Reg	Property	
has a degree	Deg	Property	
has a start date	StrtDt	Property	
has a hiring contract	HiCnt	Property	
is a faculty member	Fac	Class	{HiCnt, FacSlry, StrtDt}
has a faculty base salary	FacSlry	Property	
is an instructor	Ins	Class	{InsCnt, Slry, Crs}
has an instructors contract	InsCnt	Property	
has a course based salary	CbSlry	Property	
has a course to teach	Crs	Property	
is a graduate student	Grd	Property	
is a faculty instructor	FacIns	Category	{Fac, Ins}
is a graduate instructor	GrdIns	Category	{Grd, Ins}

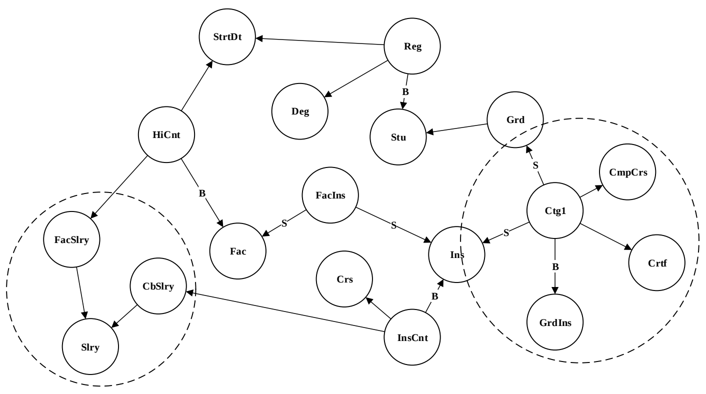


Fig. 6. Updated attribute lattice

For example, assume for a graduate student, *'completed all courses' (CmpCrs)* and *'has teaching certificate' (Crtf)* are two prerequisites to be eligible for being an instructor. This new information about *'is a graduate instructor'* changes its semantic neighbourhood and, consequently, changes the type of *{GrdIns}* from category to class. Instances that possess *{GrdIns}* or, equally, possess *{Grd}* and *{Ins}* at the same time, also possess *'completed all courses' (CmpCrs)* and *'has teaching certificate' (Crtf)*.

{Grd} and {Ins} → {CmpCrs, Crtf} or

{GrdIns} → {CmpCrs, Crtf}

As mentioned earlier, higher-level attributes can be used for semantic data integration. For instance, *'Has salary' (Slry)* is a potential higher-level attribute for

{CbSlry} and *{FacSlry}*. Fig.6 depicts the above-discussed type change and higher-level attributes (updated attributes are surrounded by dotted circles.)

4 Discussion and future directions

Resolving semantic data heterogeneity has been an active area of research in several disciplines for the past three decades. In this paper, using the premise that data semantics is inherently tied to data itself rather than schemas and borrowing principles from ontology and human cognition, we offer a new approach for representing data. We propose a formalism that represents subsumption relationships among attributes, leading to a lattice structure. In this paper, we defined the notion of the attribute lattice, its components, construction rules, and validation rules. We then illustrate these concepts with an example. A key contribution of the attribute lattice is its ***semantic relativism*** – whether a node represents a class or property depends entirely on its semantic neighbourhood – the pattern of incoming and outgoing arcs and nodes connected to these arcs. This uniformity of representation affords much greater flexibility in viewing information (multiple perspectives can co-exist), which in turn supports integration.

We envision several core uses for attribute lattices. First, as alluded to above lattices provide a semantic foundation for data integration. In traditional approaches, an impediment to data integration is structural heterogeneity between independent data sources. Known sources of heterogeneity include: what is an attribute in one source may be a class in another; concepts can be expressed at different levels of granularity in different sources; and the equivalence of attributes in different sources may be complex. In all cases, the definition of a node's semantics as the neighborhood of nodes to which it is connected provides a uniform foundation for resolving heterogeneity among data sources.

Second, attribute lattices are well-suited for analysis. Given a lattice, it is possible to automatically analyze it to determine which nodes are classes, and which inferences can be made using these classes. This analysis can be incorporated in a tool for lattice visualization (currently being developed) that enables users of data to view the data structure from various perspectives, thereby contributing to gaining insights from data. We anticipate this approach will be useful in exploring large data sets.

In this paper, we have focused on attribute lattice definition and rules. A unified attribute lattice-based data integration framework, methods for integrating lattices, possible challenges during integration, and evaluation of the effectiveness of lattice integration constitute opportunities for future research.

References

1. Noy, N.F., *Semantic integration: a survey of ontology-based approaches.* ACM Sigmod Record, 2004. **33**: p. 65–70.
2. Rahm, E. and P.A. Bernstein, *A survey of approaches to automatic schema matching.* The VLDB Journal, 2001. **10**: p. 334-350.
3. Dong, X.L. and D. Srivastava. *Big data integration*. in *2013 IEEE 29th International Conference on Data Engineering (ICDE)*. 2013.

4. Uschold, M. and M. Gruninger, *Ontologies and semantics for seamless connectivity.* ACM SIGMod Record, 2004. **33**: p. 58-64.
5. Haas, L., *Beauty and the beast: The theory and practice of information integration*, in *Database Theory–ICDT 2007*. 2007, Springer. p. 28–43.
6. Parsons, J. and Y. Wand, *Emancipating instances from the tyranny of classes in information modeling.* ACM Transactions on Database Systems (TODS), 2000. **25**: p. 228–268.
7. Wand, Y. and R. Weber, *On the ontological expressiveness of information systems analysis and design grammars.* Information Systems Journal, 1993. **3**: p. 217–237.
8. Bunge, M., *Treatise on Basic Philosophy: Ontology I: The Furniture of the World*. 1977, Dordrecht: Springer Netherlands.
9. Parsons, J. and Y. Wand, *Attribute-based semantic reconciliation of multiple data sources*, in *Journal on Data Semantics I*. 2003, Springer. p. 21–47.
10. Chen, T. and J. Parsons. *Using Property Precedence to Enhance The Effectiveness of Queries of Unstructured Data*. in *Workshop on Information Technologies and Systems*. 2008.
11. Parsons, J., *An Experimental Study of the Effects of Representing Property Precedence on the Comprehension of Conceptual Schemas*.* Journal of the Association for Information Systems, 2011. **12**: p. 401.
12. Shvaiko, P. and J. Euzenat, *A survey of schema-based matching approaches*, in *Journal on data semantics IV*. 2005, Springer. p. 146-171.
13. Rahm, E., *Towards large-scale schema and ontology matching*, in *Schema matching and mapping*. 2011, Springer. p. 3-27.
14. Wand, Y. and R. Weber, *An Ontological Model of an Information System.* IEEE Transactions on Software Engineering, 1990. **16**: p. 1282-1292.
15. Lakoff, G., *Women, fire, and dangerous things*. 1987: Chicago: University of Chicago Press.
16. Parsons, J. and Y. Wand, *Choosing classes in conceptual modeling.* Communications of the ACM, 1997. **40**: p. 63–69.
17. Parsons, J. and Y. Wand, *Using cognitive principles to guide classification in information systems modeling.* MIS Quarterly, 2008: p. 839–868.
18. Bergamaschi, S., S. Castano, and M. Vincini, *Semantic integration of semistructured and structured data sources.* ACM Sigmod Record, 1999. **28**: p. 54-59.
19. Batini, C., M. Lenzerini, and S.B. Navathe, *A Comparative Analysis of Methodologies for Database Schema Integration.* ACM Comput. Surv., 1986. **18**: p. 323–364.
20. Doan, A. and A.Y. Halevy, *Semantic integration research in the database community: A brief survey.* AI magazine, 2005. **26**: p. 83.
21. Doan, A., N.F. Noy, and A.Y. Halevy, *Introduction to the special issue on semantic integration.* ACM Sigmod Record, 2004. **33**: p. 11-13.
22. Lenzerini, M. *Data integration: A theoretical perspective*. 2002. ACM.
23. Clifton, C., E. Housman, and A. Rosenthal, *Experience with a combined approach to attribute-matching across heterogeneous databases*, in *Data Mining and Reverse Engineering*. 1998, Springer. p. 428-451.
24. Doan, A., et al., *Learning to match ontologies on the semantic web.* The VLDB Journal—The International Journal on Very Large Data Bases, 2003. **12**: p. 303-319.

25. Kang, J. and J.F. Naughton. *On schema matching with opaque column names and data values*. 2003. ACM.
26. Wache, H., et al. *Ontology-based integration of information-a survey of existing approaches*. 2001. Citeseer.
27. Berners-Lee, T., J. Hendler, and O. Lassila, *The Semantic Web.* 2001.
28. Berners-Lee, T., *Linked data-design issues.* URL http://www.w3.org/DesignIssues/LinkedData.html, 2006. **10**: p. 11.
29. Heath, T. and C. Bizer, *Linked data: Evolving the web into a global data space.* Synthesis lectures on the semantic web: theory and technology, 2011. **1**(1): p. 1-136.
30. Berners-Lee, T., R.T. Fielding, and L. Masinter, *Uniform resource identifier (URI): Generic syntax.* 2005.
31. Consortium, W.W.W., *RDF 1.1 concepts and abstract syntax.* 2014.
32. Bizer, C., T. Heath, and T. Berners-Lee, *Linked Data: The Story So Far.* 2011: p. 205-227.
33. Brickley, D. and R. Guha, *RDF schema 1.1. W3c recommendation, W3C*. 2014.
34. Fensel, D., et al., *OIL: An ontology infrastructure for the semantic web.* IEEE intelligent systems, 2001. **16**(2): p. 38-45.
35. Connolly, D., et al., *DAML+OIL (march 2001) reference description.* 2001.
36. McGuinness, D.L. and F. Van Harmelen, *OWL web ontology language overview.* W3C recommendation, 2004. **10**(10): p. 2004.
37. Hitzler, P., et al., *OWL 2 web ontology language primer.* W3C recommendation, 2009. **27**(1): p. 123.
38. Horrocks, I., P.F. Patel-Schneider, and F. van Harmelen, *From SHIQ and RDF to OWL: the making of a Web Ontology Language.* Web Semantics: Science, Services and Agents on the World Wide Web, 2003. **1**(1): p. 7-26.
39. Ghidini, C. and L. Serafini. *Reconciling concepts and relations in heterogeneous ontologies*. in *ESWC*. 2006. Springer.
40. Omelayenko, B. *RDFT: A mapping meta-ontology for business integration*. in *Proc. of the Workshop on Knowledge Transformation for the Semantic Web at the 15th European Conference on Artificial Intelligence (KTSW2002)*. 2002.
41. Šváb-Zamazal, O. and V. Svátek. *Towards ontology matching via pattern-based detection of semantic structures in owl ontologies*. in *Proceedings of the Znalosti Czecho-Slovak Knowledge Technology conference*. 2009.
42. Barrasa Rodríguez, J., Ó. Corcho, and A. Gómez-Pérez, *R2O, an extensible and semantically based database-to-ontology mapping language.* 2004.
43. Dragut, E. and R. Lawrence, *Composing mappings between schemas using a reference ontology.* On the Move to Meaningful Internet Systems 2004: CoopIS, DOA, and ODBASE, 2004: p. 783-800.
44. Lammari, N., I. Comyn-Wattiau, and J. Akoka, *Extracting generalization hierarchies from relational databases: A reverse engineering approach.* Data & Knowledge Engineering, 2007. **63**(2): p. 568-589.
45. Parsons, J., *An information model based on classification theory.* Management Science, 1996. **42**: p. 1437–1453.