D W Q

Foundations of **D**ata **W**arehouse **Q**uality

National Technical University of Athens (NTUA)
Informatik V & Lehr- und Forschungsgebiet Theoretische Informatik (RWTH)
Institute National de Recherche en Informatique et en Automatique (INRIA)
Deutsche Forschungszentrum für künstliche Intelligenz (DFKI)
University of Rome «La Sapienza» (Uniroma)
Istituto per la Ricerca Scientifica e Tecnologica (IRST)

D. Michaeli, W. Nutt, Y. Sagiv

Classification Rules for Semi-Structured Data

Proc. of the 1997 International Workshop on Description Logics (DL-97)

Gif-sur-Yvette, France, September 1997.

DWQ : ESPRIT Long Term Research Project, No 22469
Contact Person : Prof. Yannis Vassiliou, National Technical University of Athens,
15773 Zographou, GREECE Tel +30-1-772-2526 FAX: +30-1-772-2527, e-mail: yv@cs.ntua.gr

Classification Rules for Semistructured Data

David Michaeli[*] and **Werner Nutt**[†] and **Yehoshua Sagiv**[‡]

1 Introduction

Semistructured data have recently become an important research topic in databases. So far, most research has concentrated on data models and query languages [AQM+96, BDHS96, Abi97].

Interestingly, semistructured data models are based on a "world view" that is very similar to the one underlying Description Logics (DL's). A semistructured database essentially consists of objects, which are linked to each other by attributes. In this paper, we want to exploit this similarity to apply reasoning techniques from description logics to semistructured data.

Semistructured models are intended to capture data that are not intentionally structured, that are structured heterogeneously, or that evolve so quickly that the changes cannot be reflected in the structure. A typical example is the World-Wide Web with its HTML pages, text files, bibliographies etc. Another example are biological databases that are often realized as files, but users want to access them in an integrated fashion.

By their very nature, semistructured data do not come with a conceptual schema. However, adding to them a rich conceptual model is beneficial, since it would make them more accessible to users. This is especially important, since semistructured data are often accessed in an "explorative" or "browse mode," i.e., users not only query the data to find a particular piece of information, but also pose queries with the goal of having a better understanding of what information is available.

We propose in this paper to build a layer of classes on top of a semistructured data model that is an abstraction of the OEM model. The classes are defined by rules and populated by computing a minimal fixpoint. We consider two semantics for such rules, which we call "strong" and "weak" semantics. Under the strong semantics, classes are only populated by the rules, while under weak semantics users are allowed to arbitrarily insert objects into a class.

Our main results are that (1) class containment under strong semantics can be reduced to subsumption in description logics with the μ-operator, and that (2) class containment under weak semantics can be reduced to subsumption w.r.t. a set of inclusion axioms under first-order semantics.

2 A Simple Semistructured Data Model

We introduce a simple semistructured data model in the spirit of OEM [AQM+96], which we call $\mathcal{SSD}$. We assume that there is a set of *object names* (denoted as a, b) and a set of *attribute names* (denoted as P, Q).

An $\mathcal{SSD}$-database $\mathcal{I}$ consist of a, not necessarily finite[1] *set of objects*, denoted as $\Delta^{\mathcal{I}}$, where each attribute P is interpreted as a binary relation $P^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$, and each object name a as an object $a^{\mathcal{I}}$, such that distinct names correspond to distinct objects. Thus, a database can be seen as a directed graph, where each edge is labeled by an attribute name and some of the nodes have an object name. The named nodes serve as entry points to the database.

The full $\mathcal{SSD}$ data model includes also *types* like *integer*, *real*, *string*, *gif*, or *java*. Elements of types are called *value objects.* The domain of every database contains also the value objects of all types. In a database, attribute edges can end in a value object, but there are no outgoing edges from a value object.

Each type comes with a syntax for expressing *ranges.* For example, for *integer* and *real* the ranges are finite unions of intervals. Range expressions are closed under conjunction, disjunction, and complements. We assume

[*]The Hebrew University, Jerusalem 91904, Israel, davidm@cs.huji.ac.il

[†]Currently at The Hebrew University, Jerusalem 91904, Israel, nutt@cs.huji.ac.il

[‡]The Hebrew University, Jerusalem 91904, Israel, sagiv@cs.huji.ac.il

[1]Contrary to common database theory, we do not require databases to be finite. The databases that we want to describe by the $\mathcal{SSD}$ model may be so voluminous (e.g., all pages of the WWW) that from a practical point of view they are infinite.

$$
\begin{aligned}
\mathit{AdvCourse} &\leftarrow \mathit{Seminar} && (1)\\
\mathit{AdvCourse} &\leftarrow \mathit{Course} \\
&\quad \sqcap \exists \mathit{prerequisite}.\mathit{AdvCourse} && (2)\\
\mathit{BACourse} &\leftarrow \mathit{Course} \\
&\quad \sqcap \neg \mathit{AdvCourse} \\
&\quad \sqcap \forall \mathit{prerequisite}.\mathit{BACourse} && (3)
\end{aligned}
$$

Figure 1: Classification rules for a university database

that satisfiability of range expressions can be decided in polynomial time.

From these definitions, it is obvious that as mathematical objects, $\mathcal{SSD}$-databases and DL-interpretations are the same. The integration of types into our model is similar to the integration of "concrete domains" into DL's [BH91].

Note that, in most cases a semistructured database exists in an application only as a virtual database. In other words, a model like $\mathcal{SSD}$ provides a conceptual view of data that may be stored in an arbitrary way.

3 Classes and Classification Rules

With the $\mathcal{SSD}$ model, it is only possible to describe a *flat universe*: the only information about an object that we can represent consists of edges emanating from the object and edges from other objects leading to the object. In addition to such a structural description, we would like to distinguish one object from another according to the kinds of information they store. For example, we would like to express that one object represents a student while another one represents a course.

We propose to add a layer of *classes* on top of $\mathcal{SSD}$. In contrast to object-oriented databases, we cannot expect that the class membership of objects is given explicitly: objects are not created as members of classes. However, the *structure* of an object may reveal that it belongs to a class. For instance, we may recognize an object as a restaurant, because it appears in a restaurant guide and has a chef, or because it has an address, a menu, and the value of the attribute *no_of_stars* is at least 1.

We express such sufficient conditions for class membership by *classification rules*. Classes are populated by applying the rules to the objects in the database. The rules may be recursive, so that it is not sufficient to apply the rule set to an object just once.

3.1 Class Expressions

We extend our data model by class expressions to describe sets of objects. We suppose that, in addition to attribute names, object names, and range expressions, there is an infinite sets of *classes* (denoted as C, D).

For an attribute P, both P and P^{-1} are *attribute expressions*. We denote attribute expressions with the letter R.

Class expressions are built from classes, constants, and attribute expressions, using constructors as in DL's. Classes and range expressions are *basic* class expressions. The *universal* class $\top$, the *empty* class $\bot$, *cardinality restrictions* (i.e., $(\geq n\, R)$ and $(\leq n\, R)$), and *singletons* (i.e., $\{a\}$) are *flat* class expressions. A class expression is *elementary* if it is basic, flat, or of one of the forms $\neg B$, $\exists R.B$, or $\forall R.B$, where B is a basic class expression. For simplicity, we assume that only elementary expressions are allowed to appear as conjuncts in the body of a rule. If E and F are class expressions, then $E \sqcap F$, $E \sqcup F$, $\neg E$, $\exists R.E$, and $\forall R.E$ are *nested* class expressions.

To give a semantics to class expressions, we consider all databases $\mathcal{I}$ that interpret every class C as a set of objects $C^{\mathcal{I}}$. Given such a database, any expression E is interpreted as a set $E^{\mathcal{I}}$, as is done in DL's.

3.2 Classification Rules

Classification rules specify sufficient conditions for class membership. They have the form

$$C \leftarrow E_1 \sqcap \cdots \sqcap E_n,$$

where C is a class, and the conjuncts $E_1, \ldots, E_n$ are elementary class expressions. Figure 1 shows a fragment of a set of classification rules for a semistructured university database. The rules for *Seminar* and *Course* are missing.

A set of classification rules may be recursive. In order to guarantee a unique result of the evaluation of rules, we have to restrict the use of negation in the body of rules. To this end, we adapt the notion of stratification, known from Logic Programming, to our setting.

Let $\mathcal{R}$ be a set of classification rules. The *dependency graph* of $\mathcal{R}$ has as nodes the classes appearing in $\mathcal{R}$. There is an edge from C to D if there is a rule in $\mathcal{R}$ with head C such that D occurs in a conjunct of the body. An edge from C to D is a *negative edge* if there is a rule with head C such that $\neg D$ is a conjunct of the body.

We say that C *depends* on D (w.r.t. $\mathcal{R}$) if there is a path from C to D in the dependency graph. We say that C *depends negatively* on D if there is a path from C to D that contains a negative edge. The rule set $\mathcal{R}$ is *stratified* if no class depends negatively on itself.

If $\mathcal{R}$ is stratified, then the classes appearing in $\mathcal{R}$ can be organized into a hierarchy of strata $\mathcal{C}_0$, $\mathcal{C}_1, \ldots, \mathcal{C}_n$: the stratum $\mathcal{C}_0$ contains the classes that do not depend negatively on any class; the stratum $\mathcal{C}_{i+1}$ contains those that depend negatively only on classes in $\mathcal{C}_i$ or a lower stratum. The strata of classes induce strata of rules: the stratum $\mathcal{R}_i$ of $\mathcal{R}$ contains those rules whose head is in $\mathcal{C}_i$.

In the example in Figure 1, *AdvCourse* depends on itself, on *Seminar*, and on *Course*, while *BACourse* depends on itself and on *Course*, and depends negatively on *AdvCourse*. Thus, *Course* and *AdvCourse* belong to the stratum $\mathcal{C}_0$, while *BACourse* belongs to the stratum $\mathcal{C}_1$.

Our definition of stratification slightly extends the one known from Logic Programming, since it allows for rules of the form $C \leftarrow \forall R.C$ that, when translated, result in non-stratified logic programs.

4 Semantics of Rules

The semantics of a rule set $\mathcal{R}$ is given by the databases that are models of $\mathcal{R}$.

It is straightforward to define first-order models: $\mathcal{I}$ is a *first-order model* of $\mathcal{R}$ if for each rule $C \leftarrow E$ we have that $E^{\mathcal{I}} \subseteq C^{\mathcal{I}}$, where E is a conjunction of elementary class expressions. However, first-order models do not reflect the intuition that the extension of classes is computed by applying the rules.

On the level of computation, we distinguish between a strong and a weak procedural semantics. To obtain a *strong model* we start with a database that interprets only attributes and object names—we call such a database an *initial database.* We then compute the extension of the classes as in Logic Programming: first of those in $\mathcal{C}_0$, as a least fixpoint of the rules in $\mathcal{R}_0$, and then iteratively of those in each $\mathcal{C}_{i+1}$, as a least fixpoint of $\mathcal{R}_{i+1}$ where the interpretation of the classes in the lower strata has been obtained by the preceding computation.

To obtain a *weak model,* we proceed similarly, except that we allow users—or some other magic processes—to enter objects into classes before rules are applied. We can formalize this by assuming that for each class C there is a *shadow class* $\tilde{C}$. (By contrast, we will sometimes refer to the original classes as *foreground classes.*) The shadow class $\tilde{C}$ stands for the objects entered arbitrarily into C. We extend the rule set $\mathcal{R}$ to a rule set $\tilde{\mathcal{R}}$ by adding a rule $C \leftarrow \tilde{C}$ for each class C occurring in $\mathcal{R}$. The set $\tilde{\mathcal{R}}$ is called the *weakening* of $\mathcal{R}$. Obviously, if $\mathcal{R}$ is stratified, so is $\tilde{\mathcal{R}}$.

Then every weak model is obtained by first extending an initial database $\mathcal{I}$ to a database $\mathcal{I}'$ that also interprets the shadow classes as subsets of the domain (but not the foreground classes), then computing the extensions of the foreground classes by applying the rules of $\tilde{\mathcal{R}}$, similarly to applying $\mathcal{R}$ under the strong semantics, and finally forgetting the interpretation of the shadow classes.

In order to be able to reason about rules, we give declarative characterizations of both strong and weak semantics in terms of DL's. This will allow us later on to apply DL techniques to determine containment of classes.

4.1 Strong Semantics

Our goal is to translate a stratified set of rules $\mathcal{R}$ into a complex fixpoint terminology $\mathcal{T}_{\mathcal{R}}$ in the sense of Schild [Sch94], such that the strong models of $\mathcal{R}$ and the models of $\mathcal{T}_{\mathcal{R}}$ coincide.

A *terminology* $\mathcal{T}$ is a set of equations of the form $C \doteq E$, such that for any C there is no more than one equation with C on the left hand side. A class C is said to be *defined* in $\mathcal{T}$ if it occurs on the left hand side of some equation. A terminology is *syntactically positive* if no defined class occurs in the scope of a negation. Syntactically positive terminologies are guaranteed to have both least and greatest fixpoint models. A *least fixpoint* (lfp) *terminology* is an expression of the form $\mu\mathcal{T}$, where $\mathcal{T}$ is a syntactically positive terminology. The models of $\mu\mathcal{T}$ are the least fixpoint models of $\mathcal{T}$.

A terminology $\mathcal{T}$ *depends negatively* on a terminology $\mathcal{T}'$ if there is a defined class in $\mathcal{T}'$ that occurs in the scope of a negation in $\mathcal{T}$. A *complex fixpoint terminology* (cft) is a sequence $\langle \mu\mathcal{T}_0, \ldots, \mu\mathcal{T}_n \rangle$ of fixpoint terminologies, such that each $\mathcal{T}_j$ can depend negatively only on $\mathcal{T}_i$'s with $i < j$. An interpretation (or a database, in our terms) is a model of a cft if it is a model of each member of the cft.

We associate a terminology $\mathcal{T}_{\mathcal{R}}$ with a given set of rules $\mathcal{R}$ as follows: for each class C appearing as the head of a rule, $\mathcal{T}_{\mathcal{R}}$ contains the equation $C \doteq E_1 \sqcup \cdots \sqcup E_k$, where $E_1, \ldots, E_k$ are the bodies of all the rules in $\mathcal{R}$ with head C.

The translation of a stratified set of rules $\mathcal{R}$ into a cft $\mathcal{T}^{\mu}_{\mathcal{R}}$ proceeds in two steps. Let $\mathcal{R}_0, \ldots, \mathcal{R}_n$ be the strata of $\mathcal{R}$. We first translate each $\mathcal{R}_i$ into the terminology $\mathcal{T}_{\mathcal{R}_i}$. Because of the properties of strata, $\mathcal{T}_{\mathcal{R}_i}$ is syntactically positive. Then, let $\mathcal{T}^{\mu}_{\mathcal{R}} := \langle \mu\mathcal{T}_{\mathcal{R}_0}, \ldots, \mu\mathcal{T}_{\mathcal{R}_n} \rangle$. Since $\mathcal{R}$ is stratified, $\mathcal{T}^{\mu}_{\mathcal{R}}$ is a cft.

Identifying $\mathcal{SSD}$ databases and DL interpretations, we can easily characterize strong models as fixpoint models.

Theorem 4.1 (Strong Semantics and Fixpoint Semantics) *Let $\mathcal{R}$ be a stratified set of classification rules. Then the strong models of $\mathcal{R}$ and the models of $\mathcal{T}^{\mu}_{\mathcal{R}}$ are the same.*

4.2 Weak Semantics

The goal of this subsection is to show that weak models and first-order models of a stratified set of rules are the same. To this end, we define the weakening $\tilde{\mathcal{R}}$ of a set of rules $\mathcal{R}$. We relate the weak models of $\mathcal{R}$ to fixpoint models of $\tilde{\mathcal{R}}$, and then show how fixpoint models of $\tilde{\mathcal{R}}$ can be turned into first-order models of $\mathcal{R}$ and vice versa.

Recall that the weakening $\tilde{\mathcal{R}}$ of a set of classification rules $\mathcal{R}$ is obtained by adding to $\mathcal{R}$ the rule $C \leftarrow \tilde{C}$. Consequently, whenever there is an equation $C \doteq E$ in $\mathcal{T}^{\mu}_{\mathcal{R}}$, there is a corresponding equation $C \doteq \tilde{C} \sqcup E$ in $\mathcal{T}^{\mu}_{\tilde{\mathcal{R}}}$.

If $\mathcal{I}$ is a database that interprets foreground classes, but not shadow classes, then $\tilde{\mathcal{I}}$ is the extension of $\mathcal{I}$ that interprets each shadow class similarly to the corresponding foreground class, i.e., $\tilde{C}^{\tilde{\mathcal{I}}} = C^{\mathcal{I}}$. If $\mathcal{J}$ is a database that interprets foreground and shadow classes, then $\bar{\mathcal{J}}$ is obtained from $\mathcal{J}$ by forgetting the interpretation of the shadow classes. Note that $\bar{\tilde{\mathcal{I}}} = \mathcal{I}$.

By extending and forgetting, we can switch back and forth between the weak models of a rule set $\mathcal{R}$ and the fixpoint models of the weakening $\tilde{\mathcal{R}}$.

Proposition 4.2 (Weak Semantics and Fixpoint Semantics) *Let $\mathcal{R}$ be a stratified set of classification rules and $\tilde{\mathcal{R}}$ its weakening.*

1. *If $\mathcal{I}$ is a weak model of $\mathcal{R}$, then $\tilde{\mathcal{I}}$ is a model of $\mathcal{T}^{\mu}_{\tilde{\mathcal{R}}}$.*
2. *If $\mathcal{J}$ is a model of $\mathcal{T}^{\mu}_{\tilde{\mathcal{R}}}$, then $\bar{\mathcal{J}}$ is a weak model of $\mathcal{R}$.*

Similarly, we can switch back and forth between fixpoint and first-order models.

Proposition 4.3 (Fixpoint Semantics and First-Order Semantics) *Let $\mathcal{R}$ be a set of classification rules and $\tilde{\mathcal{R}}$ its weakening.*

1. *If $\mathcal{J}$ is a model of $\mathcal{T}^{\mu}_{\tilde{\mathcal{R}}}$, then $\mathcal{J}$ is a first-order model of $\tilde{\mathcal{R}}$.*
2. *If $\mathcal{J}$ is a first-order model of $\tilde{\mathcal{R}}$, then $\bar{\mathcal{J}}$ is a first-order model of $\mathcal{R}$.*
3. *If $\mathcal{I}$ is a first-order model of $\mathcal{R}$, then $\tilde{\mathcal{I}}$ is a model of $\mathcal{T}^{\mu}_{\tilde{\mathcal{R}}}$.*

Combining Propositions 4.2 and 4.3, we conclude the equivalence of weak and first-order semantics.

Theorem 4.4 (Weak Semantics and First-Order Semantics) *Let $\mathcal{R}$ be a stratified set of classification rules. Then the weak models of $\mathcal{R}$ and the first-order models of $\mathcal{R}$ are the same.*

5 Reasoning about Classification Rules

Every set of classification rules determines a *containment hierarchy* on the classes that appear as rule heads: one class contains the other if in every model it is interpreted as a superset of the set denoted by the other one. We give precise definitions of containment and other properties of classes. Then we show that in the case of strong semantics they can be determined by reasoning in a DL with fixpoints, and in the case of weak semantics they can be determined by reasoning about inclusion axioms in a suitable DL.

5.1 Reasoning Tasks

Since we distinguish between strong and weak semantics, reasoning tasks are parameterized by the two semantics. Given a stratified set of classification rules $\mathcal{R}$, we are interested in the following properties of classes C, D:

Containment: C is *strongly* (*weakly*) *contained* in D w.r.t. $\mathcal{R}$, written $\mathcal{R} \models_s C \sqsubseteq D$ ($\mathcal{R} \models_w C \sqsubseteq D$), if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for every strong (weak) model $\mathcal{I}$ of $\mathcal{R}$;

Emptiness: C is *strongly empty,* written $\mathcal{R} \models_s C \doteq \bot$, if $C^{\mathcal{I}} = \emptyset$ for every strong model $\mathcal{I}$ of $\mathcal{R}$;

Universality: C is *strongly* (*weakly*) *universal,* written $\mathcal{R} \models_s C \doteq \top$ ($\mathcal{R} \models_w C \doteq \top$), if $C^{\mathcal{I}} = \Delta^{\mathcal{I}}$ for every strong (weak) model $\mathcal{I}$ of $\mathcal{R}$.

We did not define "weak emptiness," since weak semantics can never entail that a class is empty in all models: a user can always insert an object into a class. Strong semantics, however, can entail emptiness of a class C. If C is defined by nonrecursive rules whose bodies are not satisfiable, or if C is only defined by the recursive rule $C \leftarrow C$, then C denotes the empty set in every strong model. In the example in Figure 1, Rule 1 is needed in order to prevent *AdvCourse* from being empty. If the class *AdvCourse* were only defined by Rule 2, no object would ever satisfy the conditions in the body of the rule.

5.2 Reasoning w.r.t. Strong Semantics

We have shown that the strong models of $\mathcal{R}$ and the models of the cft $\mathcal{T}^{\mu}_{\mathcal{R}}$ are the same. Thus, reasoning about classification rules w.r.t. strong semantics amounts to reasoning about cft's.

Schild [Sch94] proved that complex fixpoint terminologies can be translated in polynomial time into terminologies with embedded fixpoints. Such terminologies differ from those introduced earlier in that expressions can contain a second-order fixpoint operator. More precisely, the syntax of expressions is enhanced by allowing for (1) set variables X, and (2) fixpoint expressions of the form $\mu X.E$, where X is allowed to appear in E. Given an interpretation of all free variables and classes appearing in a fixpoint expression, then $\lambda X.E$ defines a function that maps subsets of the domain to other subsets, and $\mu X.E$ is interpreted as the least fixpoint of this function.

DL's with boolean connectives, quantification over attributes, and fixpoint expressions are actually notational variants of the propositional μ-calculus, for which the reasoning tasks defined above are EXPTIME-complete [Sch94]. De Giacomo and Lenzerini [GL94] extended the algorithms for the μ-calculus so that they can also cope with cardinality restrictions. However, to the best of our knowledge, there are no results about languages that include also inverse attributes or singletons.

Theorem 5.1 *For classification rules without inverse attribute and without singletons, reasoning w.r.t. strong semantics is decidable and* EXPTIME*-complete.*

5.3 Reasoning w.r.t. Weak Semantics

Weak semantics are equivalent to first-order semantics. In this subsection, we will show that classification rules under first-order semantics are closely related to inclusion axioms that have been extensively studied in DL's. In fact, there is a simple duality principle by which we can translate one into the other, such that models are preserved. This will allow us to reinterpret reasoning techniques and complexity results for inclusion axioms for classification rules.

Inclusion axioms have been introduced in order to model integrity constraints in database schemas and to mimic languages for conceptual modeling. We reformulate the definitions for inclusion axioms in our terminology of classes, class expressions, and databases. An *inclusion axiom* has the form

$$C \sqsubseteq E,$$

where C is a class and E an expression. A database $\mathcal{I}$ is a *model* of $C \sqsubseteq E$ if $C^{\mathcal{I}} \subseteq E^{\mathcal{I}}$. Thus, such an axiom admits only databases where each element of C satisfies the constraints expressed by E. A set of inclusion axioms is called a *schema* (and denoted as $\mathcal{S}$). The notation $\mathcal{S} \models C \sqsubseteq D$ as well as $\mathcal{S} \models C \doteq \bot$ has the meaning one expects.

Intuitively, inclusion axioms are similar to rules, except that the implication has the opposite direction. If, instead of an axiom, we consider its logically equivalent converse $\neg C \sqsupseteq \neg E$, the implication points in the same direction as in a classification rule. However, there is a negated class in the head of the rule. We will remedy this, by enriching our syntax with symbols that stand for the complements of classes.

For each class C we introduce a *mirror class* C^{mir}. We say that C^{mir} is the *mirror of* C. We generalize mirrors to class expressions E. We construct the mirror E^{mir} of E by (1) replacing each class C occurring in E by $\neg C^{mir}$, thus obtaining some E', and (2) computing the negation normal form (nnf) of $\neg E'$. Thus, if E is in nnf, E^{mir} is obtained by replacing each class with its mirror and each constructor with its dual, i.e., intersection with union, universal with existential quantification, $\leq$-restrictions with $\geq$-restrictions, etc. We write the inverse of the mirror mapping as E^{rim}.

We use mirrors to translate rule sets into schemas and vice versa. If $C \leftarrow E$ is a rule, then $C^{mir} \sqsubseteq E^{mir}$ is its *dual inclusion axiom.* Conversely, if $C^{mir} \sqsubseteq E$ is an axiom, then $C \leftarrow E^{rim}$ is its *dual rule.* If $\mathcal{R}$ is a set of rules, then its *dual schema* $\mathcal{S}_{\mathcal{R}}$ consists of the dual axioms of the rules in $\mathcal{R}$. For a schema $\mathcal{S}$, the *dual rule set* $\mathcal{R}_{\mathcal{S}}$ is defined analogously. Figure 2 shows the dual axioms of the rules in Figure 1.

Mirroring is also reflected on the semantics level. If $\mathcal{I}$ is a database that interprets classes, then $\mathcal{I}^{mir}$ is the database that has the same domain as $\mathcal{I}$, coincides with $\mathcal{I}$ for the interpretation of attributes and object names, and is defined for mirror classes by $C^{mir^{\mathcal{I}^{mir}}} := \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$. If $\mathcal{I}'$ interprets mirror classes, then $\mathcal{I}'^{rim}$ is defined in an analogous way as the database that interprets classes as the complements of their mirrors. Through mirroring, we can describe how models of rules are turned into models of schemas and vice versa. Recall that by Theorem 4.4 there is no difference between weak models and first-order models.

$$\begin{aligned}
AdvCourse^{mir} &\sqsubseteq Seminar^{mir} \\
AdvCourse^{mir} &\sqsubseteq Course^{mir} \\
&\quad \sqcup \forall prerequisite.AdvCourse^{mir} \\
BACourse^{mir} &\sqsubseteq Course^{mir} \\
&\quad \sqcup \neg AdvCourse^{mir} \\
&\quad \sqcup \exists prerequisite.BACourse^{mir}
\end{aligned}$$

Figure 2: The dual inclusion axioms for the university rules

Proposition 5.2 (Duality of Models) *Let $\mathcal{R}$ be a set of classification rules and $\mathcal{S}$ be a schema that mentions only mirror classes. Let $\mathcal{I}$ and $\mathcal{I}'$ be databases that interpret $\mathcal{R}$ and $\mathcal{S}$, respectively. Then*

- *$\mathcal{I}$ is a (weak) model of $\mathcal{R}$ if and only if $\mathcal{I}^{mir}$ is a model of $\mathcal{S}_{\mathcal{R}}$;*
- *$\mathcal{I}'$ is a model of $\mathcal{S}$ if and only if $\mathcal{I}'^{rim}$ is a (weak) model of $\mathcal{R}_{\mathcal{S}}$.*

Since models of rules and schemas are in a 1-1-correspondence, it is straightforward to reduce reasoning about rules to reasoning about schemas and vice versa.

Proposition 5.3 (Reductions) *Let $\mathcal{R}$ be a set of classification rules and $\mathcal{S}$ be a schema that mentions only mirror classes. Then*

- *$\mathcal{R} \models_w C \sqsubseteq D$ if and only if $\mathcal{S}_{\mathcal{R}} \models D^{mir} \sqsubseteq C^{mir}$;*
- *$\mathcal{S} \models C^{mir} \sqsubseteq D^{mir}$ if and only if $\mathcal{R}_{\mathcal{S}} \models_w D \sqsubseteq C$;*
- *$\mathcal{R} \models_w C \doteq \top$ if and only if $\mathcal{S}_{\mathcal{R}} \models C^{mir} \doteq \bot$;*
- *$\mathcal{S} \models C^{mir} \doteq \bot$ if and only if $\mathcal{R}_{\mathcal{S}} \models_w C \doteq \top$.*

In order to reduce reasoning about classification rules under weak semantics to reasoning about inclusion axioms under first-order semantics we proceeded in two steps. We first showed that weak semantics and first-order semantics for rules are equivalent (Theorem 4.4). Then we showed that there is a duality between rules under first-order semantics and inclusion axioms under first-order semantics (Theorem 5.2).

Reasoning about inclusion axioms and schemas has been studied in [BDS93, CLN94, BDNS95, Cal96]. This

work can now be reinterpreted in terms of classification rules. It appears that reasoning about rules without object names is decidable [CLN94]. It is unknown, however, whether reasoning becomes undecidable when object names are added.

Theorem 5.4 *For classification rules without object names, reasoning w.r.t. weak semantics is decidable and* EXPTIME*-complete.*

There are results about the decidability of reasoning in very expressive languages, as well as a detailed analysis of the complexity of reasoning for various language fragments that allows one to identify sources of complexity. Due to a lack of space, we do not give here any further details, but refer the reader to the literature.

6 Conclusion

We are involved in an Israeli WWW project, called WebSuite, where we are responsible for the conceptual modelling and querying component. To describe the information available on the Web and to query it, we are using an OEM-like semistructured data model. The WebSuite project motivated our proposal to enrich semistructured data with classes that are populated by rules. We plan to integrate classification rules into the conceptual model and to combine them with our query language.

References

[Abi97] S. Abiteboul. Querying semi-structured data. In *International Conference on Database Theory*, January 1997.

[AQM+96] S. Abiteboul, D. Quass, J. McHugh, J. Widom, and J. Wiener. The Lorel query language for semistructured data. Technical report, The Stanford University Database Group, 1996.

[BDHS96] P. Buneman, S. Davidson, G. Hillebrand, and D. Suciu. A query language and optimization techniques for unstructured data. In *Proc. 1996 ACM SIGMOD International Conference on Management of Data*, June 1996.

[BDNS95] M. Buchheit, F.M. Donini, W. Nutt, and A. Schaerf. A refined architecture for terminological systems. Research Report RR-95-09, DFKI, Stuhlsatzenhausweg 3, D-66123 Saarbrücken, Germany, June 1995.

[BDS93] M. Buchheit, F. M. Donini, and A. Schaerf. Decidable reasoning in terminological knowledge representation systems. *Journal of Artificial Intelligence Research*, 1:109–138, 1993.

[BH91] F. Baader and P. Hanschke. A scheme for integrating concrete domains into concept languages. In J. Mylopoulos and R. Reiter, editors, *Proc. 12th International Joint Conference on Artificial Intelligence*, Sydney (Australia), August 1991. Morgan Kaufmann Publishers.

[Cal96] Diego Calvanese. Reasoning with inclusion axioms in description logics: Algorithms and complexity. In W. Wahlster, editor, *Proc. 12th European Conference on Artificial Intelligence*, pages 303–307, Budapest (Hungary), August 1996. John Wiley and Sons.

[CLN94] Diego Calvanese, Maurizio Lenzerini, and Daniele Nardi. Foundations of a unified theory for class-based representation formalisms. In Jon Doyle, Erik Sandewall, and Piero Torasso, editors, *Proc. 4th International Conference on Principles of Knowledge Representation and Reasoning*. Morgan Kaufmann Publishers, May 1994.

[GL94] G. De Giacomo and M. Lenzerini. Concept language with number restrictions and fixpoints, and its relationship with mu-calculus. In A. Cohn, editor, *Proc. 11th European Conference on Artificial Intelligence*, Amsterdam (Netherlands), August 1994. John Wiley and Sons.

[Sch94] Klaus Schild. Terminological cycles and the propositional μ-calculus. In Jon Doyle, Erik Sandewall, and Piero Torasso, editors, *Proc. 4th International Conference on Principles of Knowledge Representation and Reasoning*, Bonn (Germany), May 1994. Morgan Kaufmann Publishers.