Cross: an OWL wrapper for teasoning on relational databases

Cross: An OWL Wrapper for Reasoning on Relational Databases

Pierre-Antoine Champin[1], Geert-Jan Houben[2], and Philippe Thiran[3]

[1] LIRIS, Université Claude Bernard Lyon 1
pchampin@liris.cnrs.fr
[2] Vrije Universiteit Brussel
Geert-Jan.Houben@vub.ac.be
[3] Facultés Universitaires Notre-Dame de la Paix, Namur
pthiran@fundp.ac.be

Abstract. One of the challenges of the Semantic Web is to integrate the huge amount of information already available on the standard Web, usually stored in relational databases. In this paper, we propose a formalization of a logic model of relational databases, and a transformation of that model into OWL, a Semantic Web language. This transformation is implemented in Cross, as an open-source prototype. We prove a relation between the notion of legal database state and the consistency of the corresponding OWL knowledge base. We then show how that transformation can prove useful to enhance databases, and integrate them in the Semantic Web.

1 Introduction

One of the challenges of the Semantic Web (SW) vision is to integrate, in a machine-consumable form, the huge amount of information already available on the standard Web. The long-term goal is to allow software agents to aggregate information from heterogenous sources in order to handle complex user queries. However, a great amount of the information available on the web is stored in relational databases (RDBs). From that perspective, the Semantic Web can benefit from the abundant literature on reverse engineering [1] and data integration [2] in the field of RDBs. On the other hand, SW technologies shed a new light on those classical problems, and provide new tools and methodologies, but also new challenges to the field.

Enhancing RDBs with semantically rich languages is indeed not a new idea: description logics (DLs), that happen to be one of the foundations of SW technologies, have already been considered as a unifying formalism for conceptual data models [3,4]. However, the proposed approaches were not deployed on large-scale legacy databases, notably because conceptual models of RDBs are not always available in practice. On the other hand, there have been some efforts to bridge the gap between RDBs and SW languages, but paradoxically, they have been neglecting the reasoning issue, either focusing on the syntactical level [5] or undecidable formalisms [6].

Our goal is to draw experience from all those works to provide a sound and practical approach to integrating RDBs in the SW. That approach will make it possible 1/ to use SW technologies and tools for the benefit of database engineering and re-engineering, and 2/ to smoothly integrate legacy RDBs in the Semantic Web, possibly to perform data integration with other RDBs, native RDF data sources, or even Web Services.

In this paper, we formally define an abstraction of RDBs, focusing on their logical model, and show how this fits into OWL, an SW language based on description logics. More precisely, we prove a relation between the notion of *legal database state* and the consistency of the corresponding OWL knowledge base. We then present Cross, an open-source implementation of our approach, which introduces the notion of *semantic values*.

1.1 Running Example

Along the paper, we will use as a running example the database schema described in Figure 1, which is a subset of the information system of an imaginary university. That schema describes students, the courses they attend, and their scholarship if any. It also represents the prerequisites between courses, and which courses involve practical work (practical courses).

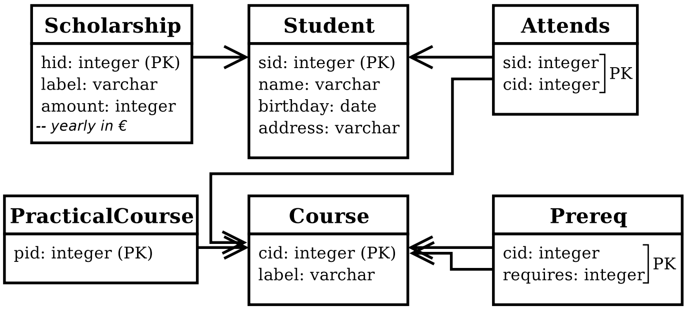


Fig. 1. An example RDB schema. PK represent primary keys. Arrows represent foreign keys.

1.2 Structure of the Paper

In Section 2 we motivate our work by a number of use cases. We then give a formal description of OWL in Section 3. Section 4 describes our formalization of RDBs logical model, that we call the ODBC model. In Section 5, we define a correspondence between both models and prove the equivalence between ODBC weak legality and OWL consistency. We present in Section 6 Cross, a working implementation of our approach. Finally we conclude and discuss some further work.

2 Motivations

OWL [7] is a knowledge representation language based on description logics and is a recomendation of the W3C. It has a well-defined formal semantics, that we will recap in Section 3. Its high expressive power allows complex inferences to be performed on OWL knowledge bases, hence the relevance of mapping relational databases to OWL. More precisely, we consider three interesting directions for such reasoning: schema reasoning and enriching, querying the data, and ensuring interoperability.

Schema reasoning and enriching. As suggested by [3], converting models into description logics brings the power of DL inference to those models, as well as additional expressiveness. The first interest of our approach w.r.t. RDBs is indeed to allow to reason about relational schemas[1], and discover implicit relations between their tables or columns.

But beyond reasoning about the schemas, mapping relational schemas to OWL allows to express *additional* constraints about them. Such constraints sometimes fit in the relational model but were omitted at design time; one can add for example in our running example axioms stating that all students must have a name. Sometimes, on the other hand, the constraints do not fit in the relational model (e.g. like cardinality constraints): we can state for example that no more than 30 students can attend a given course. We are aware that most RDB management systems (RDBMSs) allow to express that kind of constraint (e.g. using the `CHECK` or `TRIGGER` keywords from SQL), but this type of constraint is expressed in an imperative form, which makes it suited for consistency checking but unfortunately not for reasoning.

Querying the data. What we just said about the schema is, in theory, also true for the data: OWL inference engines do provide so-called $\mathcal{A}$-box reasoning services, and even elaborate query languages like SPARQL [8]. In practice, however, this is only possible with knowledge bases of a modest size, far below the size of the average corporate database. The main reason is that current inference engines must load the whole knowledge base in memory in order to reason with it. Research is being pursued on the field of distributed reasoning [9] in order to overcome this limitation, but is still at a preliminary stage.

However, we believe that an OWL mapping of the sole schema of an RDB can be used to reason about queries in order to optimize them. We will develop this point in Section 6.3.

[1] [3] actually advocate *finite model* reasonning for that purpose. Although OWL reasonners currently do not provide that kind of reasonning, we believe that "classical" OWL reasonning can nevertheless prove useful.

Interoperability. Ontologies are widely recognized as a means to achieve interoperability between heterogeneous sources of data [10]. A wealth of approaches has hence been proposed to map several relational schemas to a global ontology. However, by first translating *each* relational schema into an ontology, we also make a number of recent work on ontology aligning [11] applicable to legacy relational databases, with the opportunity to take advantage of additional knowledge enriching the schema.

3 OWL Semantics and Inferences

We recap in this section the formal semantics of OWL [7], a recommendation of the W3C to express ontologies on the SW, and present the inference services that it enables[2]. Although OWL is usually represented in XML, we favor in this paper a more compact notation which common in the DL literature [12].

Table 1. OWL constructors syntax and semantics. C and D denote concept expressions; P denotes a property symbol; n denotes a natural integer; $\#s$ denotes the cardinality of set s.

Class constructors	Syntax	Semantics
Predefined classes	$\top$	$\Delta^\mathcal{I}$
	$\bot$	$\emptyset$
Set operators	$C \sqcup D$	$C^\mathcal{I} \cup D^\mathcal{I}$
	$C \sqcap D$	$C^\mathcal{I} \cap D^\mathcal{I}$
	$\neg C$	$\Delta^\mathcal{I} \setminus C^\mathcal{I}$
Quantifiers	$\exists P.C$	$\{x \mid \exists (x,y) \in P^\mathcal{I}, y \in C^\mathcal{I}\}$
	$\forall P.C$	$\{x \mid \forall (x,y) \in P^\mathcal{I}, y \in C^\mathcal{I}\}$
Cardinality restriction	$(\leq n\, P)$	$\{x \mid \#\{y \mid (x,y) \in P^\mathcal{I}\} \leq n\}$
	$(\geq n\, P)$	$\{x \mid \#\{y \mid (x,y) \in P^\mathcal{I}\} \geq n\}$
	$(= n\, P)$	$\{x \mid \#\{y \mid (x,y) \in P^\mathcal{I}\} = n\}$
Property constructors	Syntax	Semantics
Property inverse	P^-	$\{(x,y) \mid (y,x) \in P^\mathcal{I}\}$

In OWL, a domain of interest is modeled as a set of individuals, classes denoting sets of individuals, and properties denoting binary relationships between individuals. OWL provides a number of constructors allowing to define complex classes and properties from a set of atomic classes and properties (see table 1). Features of the domain of interest are represented in an OWL knowledge base, defined hereafter.

[2] Actually, OWL has three dialects (called species): Lite, DL and Full. Only the first two of them are description logics. OWL-Full, on the other hand has an expressiveness beyond the one of DLs, but no decidable inference algorithm. In the following, mentions to OWL will only refer to its first two species. Note also that we omit on purpose some features of OWL which are not relevant to this work.

Definition 1. *An* OWL knowledge base $\mathcal{O}$ *is defined by* $\langle \mathcal{L}_\mathcal{O}, \mathcal{T}_\mathcal{O}, \mathcal{A}_\mathcal{O} \rangle$*, where:*

- $\mathcal{L}_\mathcal{O}$ *is a finite alphabet partitioned into a set* $\mathcal{C}_\mathcal{O}$ *of* class *symbols, a set* $\mathcal{P}_\mathcal{O}$ *of* property *symbols and a set* $\mathcal{O}_\mathcal{O}$ *of* individual *symbols.*
- $\mathcal{T}_\mathcal{O}$ *is a set of* axioms *as described in table 2, equivalent to the* $\mathcal{T}$*-box in the DL literature.*
- $\mathcal{A}_\mathcal{O}$ *is a set of* facts *as described in table 2, equivalent to the* $\mathcal{A}$*-box in the DL literature.*

Table 2. Syntax and semantics for OWL axioms and facts. C and D denote concept expressions; P and Q denote property expressions; i and j denote individual symbols; S denote a set of individual symbols.

	Syntax	Semantics
Class axioms	$C \sqsubseteq D$	$C^\mathcal{I} \subseteq D^\mathcal{I}$
Property axioms	$P \sqsubseteq Q$	$P^\mathcal{I} \subseteq Q^\mathcal{I}$
	transitive(P)	$\forall x, y, z \in \Delta^\mathcal{I}, (x, y) \in P^\mathcal{I} \land (y, z) \in P^\mathcal{I} \implies (x, z) \in P^\mathcal{I}$
Facts	$i : C$	$i^\mathcal{I} \in C^\mathcal{I}$
	$\langle i, j \rangle : P$	$(i^\mathcal{I}, j^\mathcal{I}) \in P^\mathcal{I}$
	$i = j$	$i^\mathcal{I} = j^\mathcal{I}$
	$i \neq j$	$i^\mathcal{I} \neq j^\mathcal{I}$
	all-different(S)	$\forall i \neq j \in S, i^\mathcal{I} \neq j^\mathcal{I}$

The semantics of an OWL knowledge base is defined by means of an interpretation $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$, consisting of an interpretation domain $\Delta^\mathcal{I}$ and an interpretation function $\cdot^\mathcal{I}$. The latter maps every individual symbol i to an element $i^\mathcal{I} \in \Delta^\mathcal{I}$, every class C to a subset $C^\mathcal{I} \subseteq \Delta^\mathcal{I}$, every property P to a relation $P^\mathcal{I} \subseteq \Delta^\mathcal{I} \times \Delta^\mathcal{I}$, while respecting the semantics of constructors as defined in table 1. An interpretation is said to satisfy a statement (axiom or fact) if it verifies the semantics of that statement as defined in table 2. An interpretation satisfying all the statements of a knowledge base $\mathcal{O}$ is said to be a *model* of that knowledge base.

A knowledge base is said to be *consistent* if it has at least one model. An axiom is said to be *entailed* by a knowledge base if every model of that knowledge base satisfies that axiom. A class C is said to be *satisfiable* under a knowledge base if that knowledge base does not entail $C \sqsubseteq \bot$, i.e. if there is at least one model $\mathcal{I}$ such that $C^\mathcal{I}$ is not empty. A class C is said to *subsume* another class D under a knowledge base if that knowledge base entails $D \sqsubseteq C$.

The problems of checking consistency, entailment, satisfiability and subsumption, are provably decidable. Several inference engines are available for OWL; we are using Pellet [13].

4 Formalizing the ODBC Model

In this section, we formalize the schema and data instance of relational databases. Although this kind of formalization is classical for conceptual data models such as the Entity-Relationship model [3,14], it has never been proposed, to the best of our knowledge, for logical data models. This makes former propositions difficult to apply for legacy databases where the conceptual model is not directly available, while logical models are. Of course, such models vary amongst the various RDBMS implementations. Nevertheless, they share a number of common notions, on which we chose to focus, and which makes it possible to port an application from one system to another. Indeed, those notions are captured by standard APIs for accessing arbitrary relational databases; this is why we named our model after one of the most popular such API: ODBC. It is not however limited to that API; other standards such as JDBC provide basically the same abstraction for relational databases, which demonstrates that the common notions they both capture are widely accepted and robust.

In the following, $tuple(S)$ denotes the set of all tuples on S (i.e. finite sequences of elements of S) of any length (including 1); if t is a tuple, $|t|$ is its length, and we note $e \in t$ if e is one of the elements of t.

Definition 2. *An* ODBC schema $\mathcal{S}$ *is defined by* $\langle \mathcal{L}_\mathcal{S}, f^T_\mathcal{S}, f^D_\mathcal{S}, \mathcal{C}^N_\mathcal{S}, f^C_\mathcal{S}, f^{ref}_\mathcal{S} \rangle$, *where:*

- $\mathcal{L}_\mathcal{S}$ *is a finite alphabet partitioned into a set* $\mathcal{T}_\mathcal{S}$ *of* table *symbols, a set* $\mathcal{C}_\mathcal{S}$ *of column symbols, a set* $\mathcal{U}_\mathcal{S}$ *of* uniqueness constraint *symbols, a set* $\mathcal{F}_\mathcal{S}$ *of* foreign key constraint *symbols and a set* $\mathcal{D}_\mathcal{S}$ *of* domain *symbols; each domain symbol* D *has an associated pre-defined basic domain* $D^{\mathcal{B}_\mathcal{D}}$. *We* do not *assume the various basic domains to be pairwise disjoint, and we suppose that, given two basic domains* d_1 *and* d_2*, the set-relation between them is known (inclusion, disjointness, etc.).*
- $f^T_\mathcal{S} : \mathcal{C}_\mathcal{S} \cup \mathcal{U}_\mathcal{S} \cup \mathcal{F}_\mathcal{S} \rightarrow \mathcal{T}_\mathcal{S}$. *Intuitively, each column, uniqueness constraint or foreign key constraint belongs to a unique table.*
- $f^D_\mathcal{S} : \mathcal{C}_\mathcal{S} \rightarrow \mathcal{D}_\mathcal{S}$. *Intuitively, each column has an associated datatype.*
- $\mathcal{C}^N_\mathcal{S} \subseteq \mathcal{C}_\mathcal{S}$ *is a subset of the column symbols. Intuitively, it denotes the columns that are required to have a value (marked NOT NULL in SQL schemas).*
- $f^C_\mathcal{S} : \mathcal{U}_\mathcal{S} \cup \mathcal{F}_\mathcal{S} \rightarrow tuple(\mathcal{C}_\mathcal{S})$. *Intuitively, each uniqueness constraint and foreign key constraint* K *applies to an ordered tuple of columns. Those columns must obviously all belong to the same table as* K*. Formally, it must hold that* $\forall c \in f^C_\mathcal{S}(K), f^T_\mathcal{S}(c) = f^T_\mathcal{S}(K)$.
- $f^{ref}_\mathcal{S} : \mathcal{F}_\mathcal{S} \rightarrow \mathcal{U}_\mathcal{S}$. *Intuitively, each foreign key references columns with a uniqueness constraint. It must hold, for every* F *in* $\mathcal{F}_\mathcal{S}$*, that it references a uniqueness constraint with the same number of columns, i.e.* $|f^C_\mathcal{S}(f^{ref}_\mathcal{S}(F))| = |f^C_\mathcal{S}(F)|$. *Note that we assume without loss of generality that the order of the columns in the foreign key matches the order of the columns in the referenced uniqueness constraint.*
- *For each table* $T \in \mathcal{T}_\mathcal{S}$*, there is at least one uniqueness constraint symbol* $U \in \mathcal{U}_\mathcal{S}$ *such that* $f^T_\mathcal{S}(U) = T$ *and* $\forall C_i \in f^C_\mathcal{S}(U), C_i \in \mathcal{C}^N_\mathcal{S}$. U *is known as the* primary key *of* T.

About the last point of that definition , we are aware that not all RDBMSs impose the existence of a primary key for every table. However, the use of primary keys is usually considered as good practice and rarely omitted.

Definition 3. *An* ODBC database state $\mathcal{B}$ *corresponding to a schema* $\mathcal{S}$ *is defined by* $\langle \Delta^{\mathcal{B}}, \cdot^{\mathcal{B}} \rangle$ *where* $\Delta^{\mathcal{B}}$ *is a non-empty finite set assumed to be disjoint from all basic domains (and sets of tuples over the basic domains), and* $\cdot^{\mathcal{B}}$ *is a function mapping:*

- *every domain* $D \in \mathcal{D}_{\mathcal{S}}$ *to the corresponding basic domain* $D^{\mathcal{B}_{\mathcal{D}}}$,
- *every table symbol* $T \in \mathcal{T}_{\mathcal{S}}$ *to a subset* $T^{\mathcal{B}}$ *of* $\Delta^{\mathcal{B}}$,
- *every column symbol* $C \in \mathcal{C}_{\mathcal{S}}$ *to a relation* $C^{\mathcal{B}} \subseteq T_C^{\mathcal{B}} \times \mathcal{V}$ *where*
  - $T_C = f_{\mathcal{S}}^T(C)$ *is the table to which* C *belongs,*
  - $\mathcal{V} = \bigcup_{D \in \mathcal{D}_{\mathcal{S}}} D^{\mathcal{B}_{\mathcal{D}}}$ *is the union of all the basic domains.*

It is furthermore assumed, for every table T*, that no two rows have the same values for the columns composing the primary key of* T*.*

Intuitively, $\Delta^{\mathcal{B}}$ can be regarded as the set of objects represented by the database; those objects are typically represented by *table rows* in the database, but a single object may be represented in several tables (i.e. an element $r \in \Delta^{\mathcal{B}}$ may belong to several $T^{\mathcal{B}}$). This allows in particular to take into account *inheritance*, which can be simulated by some patterns in relational schemas, or is even explicitly managed by some RDBMSs. Columns model attributes of those objects, hence they are represented as relations between the set of objects and the basic data domains.

The last sentence of the definition, stating that primary key constraints must be satisfied by *any* database state (rather than legal ones only) may seem misplaced. However, we need rows to be identified in some way, so we assume that this particular constraint os necessarily enforced (which, we already mentioned, is most often the case).

Note that function $\cdot^{\mathcal{B}}$ is not defined over constraint symbols (uniqueness or foreign key); indeed, those symbols do not represent elements of a database state, but merely constraints that must hold between its elements. However, for convenience, we extend the definition of $\cdot^{\mathcal{B}}$ on constraint symbols: for each $K \in \mathcal{U}_{\mathcal{S}} \cup \mathcal{F}_{\mathcal{S}}$, if $T = f_{\mathcal{S}}^T(K)$ and $f_{\mathcal{S}}^C(K) = \langle C_1, \ldots, C_k \rangle$:

$$K^{\mathcal{B}} \doteq \{(r, \langle v_1, \ldots, v_k \rangle) \mid r \in T^{\mathcal{B}}, (r, v_i) \in C_i^{\mathcal{B}}, i \in \{1, \ldots, k\}\}$$

Definition 4. *A database state* $\mathcal{B}$ *is said to be* legal for *a schema* $\mathcal{S}$ *if it satisfies the following conditions:*

- *For each* $C \in \mathcal{C}_{\mathcal{S}}$
  - $C^{\mathcal{B}} \subseteq f_{\mathcal{S}}^T(C)^{\mathcal{B}} \times f_{\mathcal{S}}^D(C)^{\mathcal{B}}$ *(range)*
  - $\forall (r_1, v_1), (r_2, v_2) \in C^{\mathcal{B}}, r_1 = r_2 \implies v_1 = v_2$ *(functionality)*
  - $\forall C \in \mathcal{C}_{\mathcal{S}}^N, \exists r \in f_{\mathcal{S}}^T(C)^{\mathcal{B}} \implies \exists (r, v) \in C^{\mathcal{B}}$ *(not null)*
- *For each* $U \in \mathcal{U}_{\mathcal{S}}$, $\forall (r_1, t_1), (r_2, t_2) \in U^{\mathcal{B}}, t_1 = t_2 \implies r_1 = r_2$ *(uniqueness)*
- *For each* $F \in \mathcal{F}_{\mathcal{S}}$, $\forall (r, t) \in F^{\mathcal{B}}$, $\exists (r', t) \in f_{\mathcal{S}}^{ref}(F)^{\mathcal{B}}$ *(reference)*

That definition of legality straightforwardly captures the constraints usually enforced by RDBMSs according to the definition of relational schemas. However, it does not have an exact correspondence in OWL (we will explain that in the next section). In the following, we will therefore need a weaker notion of legality, defined thereafter.

Definition 5. *An database state $\mathcal{B}$ is said to be* weakly legal for *a schema $\mathcal{S}$ if it satisfies all the conditions from definition 4, except for the* reference *condition.*

5 From the ODBC Model to OWL

In this section, we propose a correspondence between the ODBC model and the OWL model, and prove the equivalence between weak legality of an ODBC database state and consistency of the corresponding OWL knowledge base.

Definition 6. *Let $\mathcal{S}$ be an ODBC schema. The OWL knowledge base $\psi(\mathcal{S}) = \langle \mathcal{L}_\mathcal{O}, \mathcal{T}_\mathcal{O}, \mathcal{A}_\mathcal{O} \rangle$ is defined as follows.*

The set $\mathcal{C}_\mathcal{O}$ of class symbols contains the following elements:

- *the predefined symbols* Row*, and* Data*,*
- *for each table symbol $T \in \mathcal{T}_\mathcal{S}$, a new class symbol $\psi(T)$,*
- *for each domain symbol $D \in \mathcal{D}_\mathcal{S}$, a new class symbol $\psi(D)$.*

The set $\mathcal{P}_\mathcal{O}$ of property symbols contains for each symbol $S \in \mathcal{C}_\mathcal{S} \cup \mathcal{U}_\mathcal{S} \cup \mathcal{F}_\mathcal{S}$, a new property symbol $\psi(S)$.

The set $\mathcal{T}_\mathcal{O}$ contains the following axioms:

- *the predefined axioms:*

$$\mathsf{Row} \sqsubseteq \neg \mathsf{Data} \tag{6.1}$$
$$\top \sqsubseteq \mathsf{Row} \sqcup \mathsf{Data} \tag{6.2}$$

- *for each table symbol $T \in \mathcal{T}_\mathcal{S}$, the axiom:*

$$\psi(T) \sqsubseteq \mathsf{Row} \tag{6.3}$$

- *for each domain symbol $D \in \mathcal{D}_\mathcal{S}$, the axiom:*

$$\psi(D) \sqsubseteq \mathsf{Data} \tag{6.4}$$

- *for each domain symbol $D_1, D_2 \in \mathcal{D}_\mathcal{S}$ with $D_1^{\mathcal{B}_\mathcal{D}} \subseteq D_2^{\mathcal{B}_\mathcal{D}}$, the axiom:*

$$\psi(D_1) \sqsubseteq \psi(D_2) \tag{6.5}$$

- *for each domain symbol $D_1, D_2 \in \mathcal{D}_\mathcal{S}$ with $D_1^{\mathcal{B}_\mathcal{D}} \cap D_2^{\mathcal{B}_\mathcal{D}} = \emptyset$, the axiom:*

$$\psi(D_1) \sqsubseteq \neg\psi(D_2) \tag{6.6}$$

- *for each symbol $S \in \mathcal{C}_\mathcal{S} \cup \mathcal{U}_\mathcal{S} \cup \mathcal{F}_\mathcal{S}$ with $T = f_\mathcal{S}^T(S)$, the axioms:*

$$(\geq 1\, \psi(S)) \sqsubseteq \psi(T) \tag{6.7}$$
$$\top \sqsubseteq (\leq 1\, \psi(S)) \tag{6.8}$$

– *for each column symbol $C \in \mathcal{C}_\mathcal{S}$ with $D = f^D_\mathcal{S}(C)$, the axiom:*

$$\top \sqsubseteq \forall\psi(C).\psi(D) \tag{6.9}$$

– *for each column symbol $C \in \mathcal{C}^N_\mathcal{S}$ with $T = f^T_\mathcal{S}(C)$, the axiom:*

$$\psi(T) \sqsubseteq \exists\psi(C).\top \tag{6.10}$$

– *for each symbol $K \in \mathcal{U}_\mathcal{S} \cup \mathcal{F}_\mathcal{S}$, the axiom:*

$$\top \sqsubseteq \forall\psi(K).\mathsf{Data} \tag{6.11}$$

– *for each symbol $K \in \mathcal{U}_\mathcal{S} \cup \mathcal{F}_\mathcal{S}$ with $T = f^T_\mathcal{S}(K)$ such that all columns C_i of K are in $\mathcal{C}^N_\mathcal{S}$:*

$$\psi(T) \sqsubseteq \exists\psi(K).\top \tag{6.12}$$

– *for each uniqueness constraint symbol $U \in \mathcal{U}_\mathcal{S}$, the axiom:*

$$\top \sqsubseteq \leq 1\,\psi(U) \tag{6.13}$$

The sets $\mathcal{O}_\mathcal{O}$ of individual symbols and $\mathcal{A}_\mathcal{O}$ of facts are empty.

Intuitively, the transformation ψ maps every table row to an individual of class Row and every data value to an instance of class Data[3]. Tables are mapped to subclasses of Row, while domains are mapped to subclasses of Data. Columns are mapped to functional properties between rows and values. The constraints expressed in the schema are translated into corresponding OWL axioms.

Itis worth noting that, since axioms can not involve a *set* of properties, while relational constraints (uniqueness and foreign key) may involve several columns, ψ also creates a property for every constraint, whose values will be the tuple of values associated to that constraint. In our running example, a row of table *Attends* with values $(sid : 1, cid : 2)$ will e.g. be mapped to an individual with three properties: $\psi(sid)$ with value 1, $\psi(cid)$ with value 2, and $\psi(attends\_fk)$ with value $(1, 2)$.

Definition 7. *Let $\mathcal{B}$ be an ODBC database state corresponding to a schema $\mathcal{S}$. The OWL knowledge base $\psi(\mathcal{B}) = \langle \mathcal{L}_\mathcal{O}, \mathcal{T}_\mathcal{O}, \mathcal{A}_\mathcal{O} \rangle$ is defined as follows.*

The sets $\mathcal{C}_\mathcal{O}$ of class symbols, $\mathcal{P}_\mathcal{O}$ of property symbols and $\mathcal{T}_\mathcal{O}$ of axioms are defined according to $\psi(\mathcal{S})$ (see definition 6).

The set $\mathcal{O}_\mathcal{O}$ of individual symbols contains the following elements:

– *for each $r \in \Delta^\mathcal{B}$, a new individual symbol $\psi(r)$,*
– *for each $v \in \bigcup_{S \in \mathcal{C}_\mathcal{S} \cup \mathcal{U}_\mathcal{S} \cup \mathcal{F}_\mathcal{S}} S^{\mathcal{B}_\mathcal{D}}$, a new individual symbol $\psi(v)$,*

[3] We represent data values with OWL instances rather than concrete values (literals) because only the former allow to capture the semantics of a key (inverse-functional properties). See [15] for a discussion on that issue.

The set $\mathcal{A}_{\mathcal{O}}$ contains the following facts:

– *for $V = \{\psi(v) \mid (r, v) \in \bigcup_{S \in \mathcal{C}_{\mathcal{S}} \cup \mathcal{U}_{\mathcal{S}} \cup \mathcal{F}_{\mathcal{S}}} S^{\mathcal{B}_{\mathcal{D}}}\}$,*

$$\text{all-different}(V) \tag{7.1}$$

– *for each $T \in \mathcal{T}_{\mathcal{S}}$, for each $r \in T^{\mathcal{B}}$,*

$$\psi(r) : \psi(T) \tag{7.2}$$

– *for each $S \in \mathcal{C}_{\mathcal{S}} \cup \mathcal{U}_{\mathcal{S}} \cup \mathcal{F}_{\mathcal{S}}$, for each $(r, v) \in S^{\mathcal{B}}$,*

$$\langle \psi(r), \psi(v) \rangle : \psi(S) \tag{7.3}$$

– *for each $S \in \mathcal{C}_{\mathcal{S}} \cup \mathcal{U}_{\mathcal{S}} \cup \mathcal{F}_{\mathcal{S}}$, for each $r \in f_{\mathcal{S}}^{T}(S)^{\mathcal{B}}$ such that $\nexists (r, v) \in S^{\mathcal{B}}$:*

$$\psi(r) : (\leq 0\, \psi(S)) \tag{7.4}$$

– *for each $v \in \bigcup_{C \in \mathcal{C}_{\mathcal{S}}} C^{\mathcal{B}_{\mathcal{D}}}$, for each $D \in \mathcal{D}_{\mathcal{S}}$, if $v \in D^{\mathcal{B}_{\mathcal{D}}}$,*

$$\psi(v) : \psi(D) \tag{7.5}$$

else:

$$\psi(v) : \neg\psi(D) \tag{7.6}$$

Theorem 1. *Let $\mathcal{B}$ be an ODBC database state corresponding to a schema $\mathcal{S}$. The corresponding OWL knowledge base $\psi(\mathcal{B})$ has a model if and only if $\mathcal{B}$ is weakly legal for $\mathcal{S}$.*

The proof of this theorem is not included because of space limitation. It can be found in [16].

Limitation of the theorem. We now discuss and explain the reason of theorem 1 applying only to *weakly* legal database states, rather than strongly legal ones. We recall that a weakly legal state is not required to satisfy the "reference" constraint, i.e. that it may contain foreign keys pointing to non-existent rows. One may notice that the translation ψ of an ODBC schema into an OWL knowledge base contains no axiom about the foreign keys; it is tempting to believe that that limitation of the theorem would be alleviated by the addition of the following axiom, for all $F \in \mathcal{F}_{\mathcal{S}}$:

$$\top \sqsubseteq \forall \psi(F).(\geq 1\, \psi(f_{\mathcal{S}}^{ref}(F))^{-}\,)) \tag{7.7}$$

which states that every foreign key property must point to the value of *some* row for the corresponding uniqueness constraint. However, that axiom can be validated by $\psi(\mathcal{B})$ even if $\mathcal{B}$ does not satisfy the "reference" condition.

This is due to the fact that foreign key constraints in RDBMSs strongly rely on the so called *closed world assumption*: any information absent from the database is considered false. For example, in the schema given in Figure 1, assume a row in *Scholarship* with value 123 for *hid*, while no row in *Student* has value 123 for *sid*. This is a violation of the foreign key constraint. On the other hand, OWL reasoning is based on the *open world assumption*: any information absent from the knowledge base is considered unknown, neither true nor false. So in our example, the fact that $\mathcal{A}_{\mathcal{O}}$ does not contain an individual with value 123 for $\psi(sid)$ does *not* mean that such an individual does not exist, and since nothing prevents its existence, there is a model $\mathcal{I}$ of $\psi(\mathcal{B})$ containing that individual, even if it has no corresponding row in $\mathcal{B}$. The closed *versus* open world issue is a well-known difference between database systems and knowledge representation systems, and we will address it in more detail in the next section.

An inverse transformation. Another tempting idea is that, given an ODBC schema $\mathcal{S}$, any OWL knowledge base consistent with $\psi(\mathcal{S})$ would correspond to a weakly legal database state $\mathcal{B}$. This happens to be wrong as well, for two reasons. The first one is again related to the closed *versus* open world issue: a consistent knowledge base may be *underspecified* with regard to the schema. For example, a row may have no explicit value for a column $C \in \mathcal{C}^N_{\mathcal{S}}$; this is not inconsistent as long as it is not stated either that the row *does not* have any value for C[4].

Another problem comes from the redundancy introduced by ψ in the knowledge base: a uniqueness constraint U spanning two columns C_1 and C_2 is represented by a property of its own $\psi(U)$, independent, in the OWL knowledge base, of $\psi(C_1)$ and $\psi(C_2)$. It is therefore possible for an instance to have a tuple value for $\psi(U)$ different from its values for $\psi(C_1)$ and $\psi(C_2)$, which can of course not be represented by a database state. Furthermore, this makes it possible for two individuals to have different values for $\psi(U)$ even if their values for $\psi(C_1)$ and $\psi(C_2)$ are identical, making them artificially respect the uniqueness constraint. For those two reasons, an inverse transformation ψ^- can not exist in the general case.

6 Cross: An Implementation

In this section, we present Cross, an implementation of our approach presented before. This implementation is an open-source software, available at `http://liris.cnrs.fr/~pchampin/dev/cross`. We first stress a number of differences between the theoretical model and the actual implementation, and develop the most saillant of them: semantic values. Additionally we show the benefits of using Cross in the motivating use cases described in Section 2.

6.1 Differences to the Theoretical Model

There are three differences between the theoretical model described above and the actual implementation. In the following we describe those differences and explain why they do not affect the validity of theorem 1.

[4] A statement made by fact 7.4 for columns with null values in the database state.

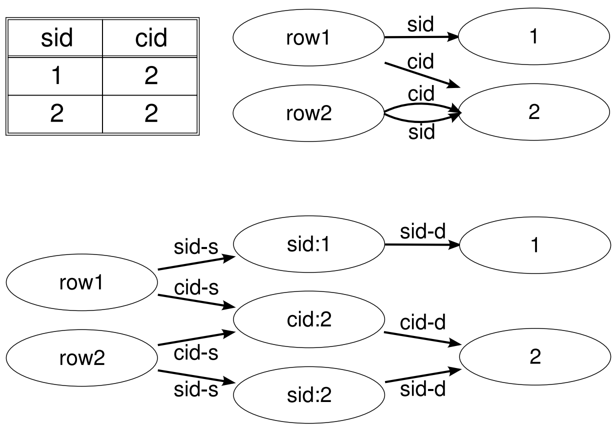
The first difference is an effort to reduce the redundancy in the OWL knowledge base. In Section 5, we remarked that transformation ψ creates redundant information by associating to every uniqueness constraint a property which is independent of the properties associated to the columns concerned by that constraint. This is useful for multi-column constraints, because the axioms 6.13 guaranteeing the uniqueness can only apply to a *single* property. On the other hand, for constraints spanning a single column, there is no need for an additional property: the property associated to the column can represent the uniqueness constraint as well, and axiom 6.13 can be applied directly to the column property. This is what the implementation does, and it does the same for properties representing foreign keys. That difference makes definitions 6 and 7 and the proof of theorem 1 a little more complex (they require to treat single-column constraints differently from multi-columns constraints) but not significantly different.

The second difference is that, for the sake of completeness, the implemented transformation includes an axiom similar to axiom 7.7, i.e. forcing foreign keys to point to an existing value. We already explained in the discussion following the proof (Section 5) that this is not sufficient to strengthen the theorem. However it does not weaken it either, because that axiom adds no real constraint to the knowledge box: it demands the existence of an individual that no other axiom generated by our approach prevents from existing. So if the knowledge base without that axiom has a model, then it also has a model with the axiom. The axiom may nevertheless prove useful in the reasoning tasks described in Section 6.3, making explicit a constraint that is actually satisfied by legal database states.

The third difference is about the representation of data. While the transformation presented in Section 5 straightforwardly creates an individual per row and an individual per data value, Cross introduces an intermediate layer of individuals, as illustrated on Figure 2. While individuals of the rightmost layer represent raw data values (in the figure: the number 1, the number 2), individuals of the new intermediate layer represent values in the *context* of a given column. We call them *semantic values*, in the manner of [17]. Indeed, the number 1 must be treated differently when it represents, e.g., a length in meters or a price in euro. In a sense, semantic values can be viewed as reifications of the arcs from the straightforward transformation: they do not provide additional information, but only express the same information in a more detailed fashion (which will prove useful in Section 6.3). As a consequence, theorem 1 still holds for the transformation with semantic values.

6.2 Dealing with Semantic Values

As we saw, Cross creates for each column C two OWL object properties. The first one, noted $\phi_s(C)$, links the individual representing the row (row individual) to the semantic value, while the second one, noted $\phi_d(C)$, links the semantic value to its data value. In the straightforward transformation, $\psi(C)$ captures all the semantics of the column, while that semantics is somewhat split into $\phi_s(C)$ and $\phi_d(C)$. For example, consider column *Scholarship.amount*; values for that



sid	cid
1	2
2	2

Fig. 2. An extract of table *Attends* (up-left), the straightforward transformation (up-right), and the Cross transformation (down) with semantic values

column represent the yearly amount of money, in Euro, received by the holder of the scholarship. In Cross, we can decide that the semantic value represents the yearly amount of money, independently of the currency; hence $\phi_s(amount)$ links a scholarship to the income it provides yearly, while $\phi_d(amount)$ links an amount of money to its value in Euro (but another property could link the same semantic value to its amount in US Dollars).

Note that our approach, though not incompatible, is different from the one proposed by [17]. The latter is to attach attributes[5] to columns and values in order to make their semantics explicit. According to their model, all values for *Scholarship.amount* would have the attributes ($Periodicity$ = '*yearly*', $Currency$ = '*Euro*'). Such elicitation requires of course a precise ontology of column attributes, which is not at all trivial. On the other hand, our approach assumes *a priori* that all columns have a distinct semantics (each column has its own "semantic value space"), and relies on human intervention to state differently, if deemed relevant. With the appropriate OWL axiom, one can indeed state that two columns happen to have the *same* semantics (e.g. two columns representing a yearly amount of money). We believe that this approach is more robust (because it assumes difference by default) and scalable (because it does not require an ontology of column attributes). Of course, should such an ontology be available, it could be used to formally document properties generated by Cross: using OWL annotations, it would be possible with an appropriate vocabulary to state that $\phi_s(amount)$ links to a yearly income as a semantic value, while $\phi_d(amount)$ links to a value in Euro as an integer. We see that the two approaches are actually complementary.

Given two columns C_1 and C_2, stating an equivalence between $\phi_s(C_1)$ and $\phi_s(C_2)$ means that values for the two columns are *commensurable*, i.e. that they

[5] The authors call them *properties*; we use the term "attribute" to avoid confusion with OWL properties.

have a common semantics (e.g. an amount of money), but that their values are not necessarily comparable (e.g. in different currencies). On the other hand, stating an equivalence between $\phi_d(C_1)$ and $\phi_d(C_2)$ means that the values are *comparable*, but not necessarily that they have the same semantics (e.g. an income and a price). Let us note that the boundary between ϕ_s and ϕ_d is not as objective as it may seem. For example, we decided above that semantic values for *Scholarship.amount* represent a *yearly* income, and that $\phi_d(amount)$ links it to its value in Euro. But we could as well have decided that the semantic value represents a *periodic* income, and that $\phi_d(amount)$ links it to its yearly value in Euro. That would make semantic values of *amount* commensurable with any other periodic income, whatever its periodicity.

6.3 Use Cases

In this section, we revisit the goals stated in Section 2 and show how Cross contributes to their achievement.

Schema reasoning and enriching. The first interest of our approach is to be able to express additional constraints on the ODBC schema and to reason about them. OWL expressiveness goes beyond the one of SQL, for example, with respect to relationships between classes: specialization, disjointness, or equivalence can easily be expressed in OWL. While some patterns in an ODBC schema can be used to simulate specialization (e.g. the primary key of *PracticalCourse* being a foreign key to *Course*), those patterns can not always be interpreted that way (see as a counterexample *Scholarship* and *Student*). Such specialization can be made explicit either directly ($\psi(PracticalCourse) \sqsubseteq \psi(Course)$) or by using columns ($\phi_s(pid) \sqsubseteq \phi_s(cid)$); the latter is preferable, because it also allows to properly identify the instances of the class[6].

Querying the data. While RDBMSs are usually capable of checking the legality of a database state, the use of an OWL inference engine to check the consistency of the corresponding knowledge base could in theory take into account additional constraints that are not known to the RDBMS. However, we already stressed the fact that it is only feasible for databases of a limited size.

Cross may however prove useful, if not to query the data, to reason about the queries themselves. It has been proved in [18] that some conjunctive queries on an OWL knowledge base can be reduced to a class expression. For that kind of queries, class satisfiability can be checked before executing, to ensure that the query can actually hold results. Furthermore, class subsumption can be used to test query containment, hence to help optimize queries by reusing cached results of containing queries [19]. The originality of using OWL reasoning for this purpose is its ability to take into account additional constraints expressed

[6] Provided also that $\phi_d(pid) \sqsubseteq \phi_d(cid)$, i.e. stating that *PracticalCourse* identifiers are special cases of *Course* identifiers. Any practical course with $pid = x$ will be recognized as having $cid = x$ as well. Since *cid* is functional, it will be identified as the course with $cid = x$.

in OWL, that would otherwise have been buried in triggers, `CHECK` constraints, or application code.

Interoperability. Interoperability has been a primary goal in the development of Cross. We argued that semantic values and the splitting of columns in two OWL properties allow fine grain comparison of column semantics (commensurable, comparable). Hence we believe that they provide a high flexibility for aligning the generated ontology with other ones. Considering an ontology about students and scholarship where US Dollar would be used instead of Euro. Without semantic values, we could only align classes representing scholarships, but not their properties. With semantic values, we can nevertheless state that the OWL property $\phi_s(amount)$ is equivalent to the corresponding property in the other ontology: they indeed bear the same general meaning. On the other hand, the fact that their numeric values can not be compared will be conveyed by the fact that $\phi_d(amount)$ would *not* be aligned.

7 Conclusion and Perspectives

In this paper, we have proposed the ODBC model, a formalization of relational databases focusing on their logic model. We have then presented a transformation of that model into OWL, a DL-based language designed for the Semantic Web. This transformation is implemented by the Cross open-source prototype, which effectively introduces the interesting notion of semantic values. We proved that the knowledge based produced by this transformation is consistent if and only if the source database state is weakly legal (i.e. legal but regarding foreign key constraints). Taking advantage of that result, we have shown how that transformation can prove useful for the purpose of analysing legacy RDBs, enhancing existing RDBs with additional constraints, and integrating them in the SW.

A first direction for further work would be to try and strengthen the theorem, to have an equivalence of OWL consistency with *full* legality, i.e. taking into account foreign keys. This could actually be done by using an expressive feature of OWL (the *oneOf* constructor, not mentioned in this paper), but would possibly make the reasoning intractable. Another solution would be to propose, in a similar way to *finite model* reasoning [3], an algorithm of *closed world* reasoning which would not be allowed to create individuals.

We also want to get more experimental results for the Cross implementation. Preliminary results[7] are encouraging: the transformation of the schema of real database (127 tables, 869 columns, 132 unicity constraints, no foreign key) took around 1.5s; the resulting ontology was loaded in Pellet in about 9s, while reasoning took about 3s. Those results seem reasonable for a quite big schema. We now plan to experiment on the use cases presented in Section 6.3 with that database and a sample of other real databases.

[7] On an Intel Core 2, 2.33GHz, with 2GB of memory.

References

1. Hainaut, J., Henrard, J., Hick, J., Roland, D., Englebert, V.: Database Design Recovery. In: Constantopoulos, P., Vassiliou, Y., Mylopoulos, J. (eds.) CAiSE 1996. LNCS, vol. 1080, pp. 272–300. Springer, Heidelberg (1996)
2. Halevy, A.Y., Ives, Z.G., Mork, P., Tatarinov, I.: Piazza: data management infrastructure for semantic web applications. In: WWW'03, pp. 556–567. ACM Press, New York (2003)
3. Calvanese, D., Lenzerini, M., Nardi, D.: Unifying Class-Based Representation Formalisms. J. Artif. Intell. Res (JAIR) 11, 199–240 (1999)
4. Levy, A.Y., Rousset, M.C.: Combining horn rules and description logics in carin. Artif. Intell. 104(1-2), 165–209 (1998)
5. Bizer, C.: D2R MAP - A Database to RDF Mapping Language. In: WWW 2003 (Posters) (2003)
6. de Laborda, C.P., Conrad, S.: Relational.OWL - A Data and Schema Representation Format Based on OWL. In: Hartmann, S., Stumptner, M. (eds.) APCCM 2005, vol. 43 of CRPIT, pp. 89–96. Australian Computer Society (2005)
7. Dean, M., Schreiber, G.: OWL Web Ontology Language. W3C Recommendation (2004), http://www.w3.org/TR/owl-ref/
8. Prud'hommeaux, E., Seaborne, A.: SPARQL Query Language for RDF. W3C Working Draft (2007), http://www.w3.org/TR/rdf-sparql-query/
9. Grau, B.C., Parsia, B., Sirin, E.: Combining OWL ontologies using ϵ-Connections. Web Semantics: Science, Services and Agents on the WWW 4(1), 40–59 (2006)
10. Wache, H., Vögele, T., Visser, U., Stuckenschmidt, H., Schuster, G., Neumann, H., Hübner, S.: Ontology-based integration of information — a survey of existing approaches. In: Stuckenschmidt, H. (ed.) IJCAI–01 Workshop: Ontologies and Information Sharing, pp. 108–117 (2001)
11. Euzenat, J.: An API for ontology alignment. In: McIlraith, S.A., Plexousakis, D., van Harmelen, F. (eds.) ISWC 2004. LNCS, vol. 3298, pp. 698–712. Springer, Heidelberg (2004)
12. The Description Logic Handbook: Theory, Implementation, and Applications. In: Baader, F., et al. (eds.) Description Logic Handbook. Cambridge University Press, Cambridge (2003)
13. Sirin, E., Parsia, B., Grau, B.C., Kalyanpur, A., Katz, Y.: Pellet: A practical owl-dl reasoner. Journal of Web Semantics (to appear)
14. Borgida, A., Lenzerini, M., Rosati, R.: Description Logics for Databases. DLHandbook [12], 462–484
15. Champin, P.A.: Representing data as resources in rdf and owl. In: Arenas, M., Hidders, J. (eds.) EROW 2007. CEUR Workshop Proceedings (January 2007), http://ceur-ws.org/Vol-229/
16. Champin, P.A., Houben, G.J., Thiran, P.: Wrapping relational databases on the semantic web. Technical Report RR-LIRIS- 2007-012 (2007), http://liris.cnrs.fr/publis/?id=2797
17. Sciore, E., Siegel, M., Rosenthal, A.: Using semantic values to facilitate interoperability among heterogeneous information systems. ACM Transaction on Database Systems 19(2), 254–290 (1994)
18. Glimm, B., Horrocks, I., Lutz, C., Sattler, U.: Conjunctive query answering for the description logic $\mathcal{SHIQ}$. In: IJCAI 2007 (2007)
19. Stuckenschmidt, H.: Similarity-based query caching. In: Christiansen, H., Hacid, M.-S., Andreasen, T., Larsen, H.L. (eds.) FQAS 2004. LNCS (LNAI), vol. 3055, pp. 295–306. Springer, Heidelberg (2004)