Exploitation of Ontology Languages for both Persistence and reasoning Purposes : Mapping PLIB, OWL and Flight ontology models.

Chimène Fankam, Yamine Aït-Ameur, Guy Pierra

EXPLOITATION OF ONTOLOGY LANGUAGES FOR BOTH PERSISTENCE AND REASONING PURPOSES

Mapping PLIB, OWL and Flight ontology models *

Chimene Fankam, Yamine Ait-Ameur, Guy Pierra
LISI / ENSMA
http://www.lisi.ensma.fr
Teleport 2 - 1 avenue Clement Ader - BP 40109 - 86961 Futuroscope Chasseneuil Cedex - FRANCE
{*chimene.fankam, yamine, pierra*} *@ensma.fr*

Keywords: Ontology model : PLIB, OWL, FLogic; Database; Mapping.

Abstract: Ontologies are seen like the more relevant way to solve data understanding problem and to allow programs to perform meaningful operations on data in various domains. However, it appears that none of the proposed models is complete enough by itself to cover all aspects of knowledge applications. In this paper, we analyse existing modeling approaches and classify them according to some revelant characteristics of knowledge modeling we have identified. Finally, this paper present transformation rules between ontology models in order to get benefits of their strengths and to allow a powerful usage of ontologies in data management and knowledge application.

1 INTRODUCTION

Nowadays, ontologies are seen as a core technology and a fundamental data model for knowledge systems. In this context, they are used in various research fields and application domains. Several ontology models using different languages or formalisms have been proposed. These models have their own domain of efficiency (semantic web, integration, knowledge sharing, database, reasoning, data exchange, etc.). It appears that, the language or the formalism used by these proposals is influenced by the target domain, by the difficulties to overcome and by the needs identified when defined. When taking into account these parameters, the efficiency and computability of each model should now be appreciated according to some specificities of the context in which it has been developed (query answering, concept modeling, instance storage, ...). However, two major problems arise: (1) people may face the difficult problem of interoperability since the constructs offered by each model are different; (2) people may encounter some difficulties to choose the right ontology model to use or to apply in practical engineering areas since the objectives of each model are different and context-dependent.

In several application domains and particularly in the engineering area in which we are involved, concepts and data are represented by classes and properties describing components with instances stored in databases. The PLIB ontology model(ISO-13584-42, 1998), initially created for electronic and mechanic components specification, catalogues storage and exchange, as well as the OntoDB(Pierra et al., 2005) ontology based database, have been defined in this area. PLIB ontologies allow to describe the knowledge shared by engineers in their design activities. In parallel, languages and models like OWL(Bechhofer et al., 2004) and OWL Flight(de Bruijn et al., 2004c) have emerged. These models target Web applications, reasoning, workflow management, integration, electronic commerce, etc. in different application domains. At this stage, we notice that each ontology model has it own domain of efficiency and its own targeted applications. Therefore, it seems necessary to allow the possibility to use each model in an integrated framework. Our work investigates the integration of PLIB and OntoDB models with other models like OWL and OWL Flight in order to allow the concurrent usage of these models. This paper shows how model transformations, using model mappings allow to get the benefits of both of them.

This paper is organized as follow. We first discuss the relevant characteristics of a modeling approach from the expressiveness and database points of view

* This work has been partly supported by the French ANR under grant ANR05RNTL02706 (e-Wok-Hub).

in section 2. Then, we briefly describe the three ontology models targeted by our work in section 3. Section 4 presents the mapping rules we have elaborated followed by the description of how these mappings are implemented in section 5. Finally, we provide some conclusions and perspectives of this work.

2 SOME RELEVANT CHARACTERISTICS OF KNOWLEDGE MODELING

This section outlines the weaknesses and strengths of various ontology-based modeling approaches from the expressiveness and database points of view. We discuss some characteristics we have judged relevant to distinguish ontology models. The choice of these characteristics is based on similar studies comparing OWL and OWL Flight(Patel-Schneider and Horrocks, 2006; de Bruijn et al., 2004b).

2.1 Strong typing

An ontology typically consists of a number of classes, relations (usually called properties) between these classes, instances and axioms. While translating an ontology schema into a database, a class is usually translated as a table while a property is translated to a column of a given type in a table which corresponds to the class in which it can be applied.

So, databases require strong typing modeling since a property is linked to one class (its domain) and is always associated with a type (its range) which can be either a data type or a reference to class in case the property defines a relationship between two classes of the ontology. Some languages like those founded on Description Logic(DL)(Baader et al., 2003) do not impose strong typing. In such languages the problem of satisfiability of concepts may arise. Thus, ontology modeling must be submitted to a rigorous analysis in order to avoid semantic errors due to unsatisfiable descriptions. For example, given the class *Person* which corresponds to a set of humans, and a property *flavor* whose domain is not explicitly specified, the OWL description *intersectionOf(Person, Restriction(flavor, value(acid)))* is unsatisfiable since the property *flavor* has no sense as a characteristic of the class *Person* although it may have a sense for another class of the ontology.

2.2 Constraint

In a model, a constraint is a logical expression that should always be valid (invariant). In many engineering area like e-commerce, constraints are important, they are used to check the consistency of the knowledge base(KB). In a KB, two types of constraint in their implicit form are considered : property value constraint (the range of the property in its definition) and property cardinality constraint(with default value one). Other constraints cannot be captured by the classical knowledge base or the model itself; it is the case for global constraints which hold on classes rather than on properties. In those application domains, the more an ontology model and the underlying formalism enable constraints specification, the more it is appreciated by the engineers. However, one should care about the evaluation of such complex constraints.

2.3 OWA vs CWA

OWA(Open World Assumption) and CWA(Closed World Assumption) are two different ways of interpreting data. CWA is typically applied in logic programming and in database applications(de Bruijn et al., 2004a). It assumes to have a complete knowledge of the world which implies that if the fact C is not a consequence of the KB, its negation is. So, CWA deals with defaults and schema-data mappings. Indeed, integrity constraints can be checked and validated and data structure are typically closed. Some critical domains of engineering like e-commerce are inherently closed worlds.

When the world is assumed to be open, the logical operator TRUE and FALSE are not interpreted as the strict negation of each other. It implies that a fact C and its negation cannot be consequences of a KB at the same time. OWA assumes incomplete information by default. The fact that C is not a consequence of the KB is not sufficient to assert that its negation is a consequence of the KB. Thus, OWA supposes that a KB can be intensionally underspecified in order to allow others to reuse or to extend it. OWA is typically used in the semantic web where the information is distributed, in science where all the answers are not yet known. In these areas, new assertions can be deduced and, equivalences between facts can be inferred.

2.4 Context modeling

Taking into account the context in which a class or a property is defined is important while modeling a domain. For example in mechanics, the resistance of a hull depends on its temperature. This dependency relationship among properties needs to be expressed at the ontology level. Another point about property is related to its value. Elaborated data type systems

are required by real-world applications; the value of the property temperature for example is meaningless if it is not associated with an unit. In such domains, a model cannot be used if it does not offer mechanisms to capture such informations.

2.5 Inheritance and instantiation

Inheritance is a powerful mechanism allowing an object to have the same characteristics as another object. Inheritance is usually declared at the class level. This relationship between classes defines a class hierarchy. In many languages(object-oriented and logics), multiple inheritance is allowed, i.e., a class may belong to more than one inheritance hierarchy. Many implementations of these languages including Java and many data structure systems like XML Schema support single inheritance only. Multiple inheritance must then be used with care because it may lead to integration problems due to the absence of explicit or direct implementation. Relational database systems like Oracle and Postgres which are more widespread and efficient than the object database systems support single inheritance only.

Instances correspond to objects or individuals described by a class. The set of instances of a class describes its extension or population. An instance may be an instance of a single class or of more than one in case of multi-instantiation. In a database, an instance is linked to one table only.

2.6 Reasoning

Reasoning tasks, like subsumption and classification can be very useful in data integration. Subsumption can assist in matchmaking classes and in finding candidate classes to link ontologies. Class membership inference and classification can be used to check whether an individual is a member of a specific class and to allow automatic migration of instances from one class to another one. Other logical characteristics like reflexivity, symmetry, ... can contribute to automatic information filling, and improve query rewriting and thus answering.

3 THREE ONTOLOGY MODELS

For our overview, we have chosen three different but overlapping ontology models : the PLIB model targeting databases applications, OWL for semantic web application and F-logic which handles procedural knowledge and addresses deductive databases and the semantic web.

3.1 PLIB ontology model

The Parts LIBrary ontology model was developed to describe technical and engineering component objects and to provide integration mechanisms. Components objects are described by means of consensual concepts of a target domain defined by their relevant properties.

3.1.1 Main characteristics

A PLIB ontology model is (Pierra, 2004) : (1) conceptual i.e., each entity and each property are completely defined. The terms (or words) used for describing concepts are only a part of their formal definitions, (2) multilingual i.e., each entry is associated with a globally unique identifier (GUI), words used in some facets may appear in many languages, (3)formal i.e., the PLIB ontology model is formally specified in EXPRESS(Spiby and Schenck, 1994) and is unambiguous, (4) modular i.e., an ontology may reference another ontology by an external reference, (5) consensual i.e., the conceptual model of the PLIB ontology reached an international consensus and has been published as an international standard (ISO13584). Most of PLIB domain ontologies are also published as international standards (e.g., IEC1380:1998).

3.1.2 Formal definition

Formally, (for a more complete description, see(Pierra, 2004)) a PLIB ontology O is defined as the 4-tuple $O : <C, P, Sub, Applic>$, where:

- C is the set of classes describing the concepts of a given domain;
- P is the set of properties describing the instances of C. It is assumed that P defines a much greater number of properties than those usually represented and valued in the instances stored in a database. Only a subset of them might be selected for a particular database;
- Sub is the subsumption function. PLIB implements two kinds of subsumption : is_a which is the usual single inheritance used to specialize a class and is_case_of which is a particular implementation where properties are not inherited but may be explicitly (and partially) imported from another class.
- $Applic : C \rightarrow 2^P$ is a map which associates to each ontology class those properties that are applicable (i.e., rigid) for each instance of this class.

3.1.3 Modeling fundamentals

The PLIB model adopts the CWA and supports other capabilities including (1) strong typing : each property is defined in the context of a class that constitutes its domain, and it has a meaning only for this class and its possible subclass(es); (2) context modeling : a property value may depend upon its evaluation context (e.g., the weight of an object depends on the gravity of the place where the object is located), the property value may be associated with a measure unit; (3) multiple points of view : an object may be characterized by one single class, and it may be represented by any number of discipline-specific ontology class, the point of view itself being represented by an ontology class; (4) it is also possible to define mathematical derivation among properties (e.g., the diameter of a circle equals its radius times 2).

3.1.4 Usage

Nowadays, the PLIB ontology model is put into practice in various domains including integration of heterogeneous data sources and data warehousing(Xuan et al., 2006). Modeling of engineering component, electronic catalogues and Semantic Web(Aklouf et al., 2003).

An efficient ontology based database architecture associated with PLIB and named OntoDB is available. Several tests(Dehainsala et al., 2007) have pointed out the effectiveness of this architecture compared to other existing ontology-based databases like RDF-Suite or Sesame.

3.2 OWL ontology model

The Ontology Web Language is a recommendation of the W3C for expressing ontologies in the semantic Web. It has been defined starting on the work on RDF(Resource Description Framework), RDFS(RDF Schema), DAML-OIL and DL. It offers reasoning capabilities over the ontology to check consistency and to determine logical consequences (inferences).

3.2.1 Main characteristics

OWL features several characteristics including (1) multiple inheritance i.e., a class may belong of several hierarchies, (2) property relaxation, unlike typical properties in object-oriented programming and frame-based ontology languages, properties in an OWL ontology are not defined as a part of the class. Indeed, OWL allows to defining a property without specifying to which class it applies or in which set it takes its values, (3) subproperty i.e., in OWL, subsumption relationship is also declared at property level, (4) sub-species i.e., the OWL language provides three increasingly expressive sublanguages, OWL Lite, OWL DL and OWL Full. Each of these sublanguages is an extension of its simpler predecessor, both in what can be legally expressed and in what can be safely concluded.

In the remainder of this paper, we are only concerned with the most well-known and most investigated specie of OWL, namely OWL DL.

3.2.2 Formal definition

OWL DL has a direct model-theoretic semantics similar to model theoretic semantics for DL. Its power consists in its reasoning support built from subsumption, classification, instantiation and satisfiability. Note that all these reasoning are usually reduced to DL (un)satisfiability.

Formally, DL languages typically have set-based model-theoretic semantics, in which a concept C is mapped to a subset of a domain Δ^I under an interpretation I using a mapping function ^{I}. Similarly, a role R is mapped to a binary relation over the domain $\Delta^I \times \Delta^I$. Equivalence of concepts is interpreted as equal subsets ($C \equiv D$ is interpreted as $C^I = D^I$), subsumption is interpreted as a subset relation ($C \sqsubseteq D$ is interpreted as $C^I \subseteq D^I$), and so on. The semantics of DL adopts the OWA.

3.2.3 Modeling fundamentals

OWL ontologies may be non-canonical ontologies because they allow derived concepts. Indeed, OWL offers a large set of constructors allowing to build: (1) descriptions specifying derived or defined classes using logical operators (union, intersection, complement), (2)restrictions, (3) logical characteristics(reflexive, symmetric, transitive) of properties, (4) relations on classes or on properties using equivalence and disjunction axioms.

3.2.4 Usage

OWL is used by applications that need to discover similar meanings represented through different descriptions. Its main application domain is the semantic web where it is used for describing concepts and terms, for information retrieval and for interconnecting web services.

3.3 F-LOGIC

Frame logic is a deductive, object-oriented knowledge information modeling language which combines the declarative semantics and expressiveness of deductive database languages with rich data modeling capabilities supported by object-oriented data models. The basic idea of F-logic is to take complex data types as in object-oriented databases and to combine them with logic and the reasoning capabilities it offers. The result is a powerful knowledge representation and query language.

3.3.1 Formal definition

F-logic has a model theoretic semantics and a sound and complete resolution based proof theory founded on first-order predicate and on frame calculus(Kifer et al., 1995). Formally, an F-logic theory is a set of formulas built using molecules. A molecule in F-logic is one of the following assertions :

- *C : D* expressing class membership (C is an instance of D),
- *C :: D* expressing subclass relationship (C is a subclass of D), or
- *C[D → E]* expressing that for the individual C, the property D has the value E.

In F-logic, the semantics is given in terms of signatures and formulas satisfaction. Molecules are associated to classes via signature expressions which specify a type constraint on the scope of a property and the type of its value. A formula (including rules) is defined by combining molecules with the usual logic connectors $\wedge$, $\vee$, $\leftarrow$, $\neg$, and quantifiers $\forall$, $\exists$.

3.3.2 Main characteristics

F-logic characteristics include : (1) Unique Name Assumption(UNA) i.e., any object in F-logic has a unique identifier (its name) which is used to reference it, (2) multiple inheritance, (3) strong typing i.e., F-logic includes the definition of properties in the same structure where the class is defined, (4) n-ary relationship, on the contrary of most ontology formalisms, where properties are binary relationships, F-logic expresses n-ary relationships; a property is called either attribute i.e., with no argument or method i.e., with arguments, (5) polymorphism as in object-oriented languages, (6) meta-modeling i.e., in F-logic, both classes and instances belong to the same domain; thus an object can be seen as a class or an instance depending of its role in a particular context, (7) presence of rules that provide additional expressiveness on top of the ontology via a flexible mechanism to add and to derive implicit knowledge to and from the ontology in efficient way.

3.3.3 Usage

F-logic is used in semantic web particularly for ontology management, semantic web services, rule-based extension of ontologies, software engineering, modeling of intelligent agents, knowledge representation in a frame-based approach, and representation of object-oriented databases.

3.3.4 OWL Flight

Having a rule language in extension of the ontology language in order to allow efficient query answering has been identified as a requirement for the semantic web. OWL Flight has been proposed with rules extension for OWL.

OWL-Flight restricts OWL DL and extends the restricted subset of OWL DL(de Bruijn et al., 2004a) with a semantic based on logic programming with F-logic semantic sugar rather than DL. It also borrows the constraint-based modeling style common in database.

OWL Flight imposes several restrictions to OWL DL. Indeed, complement classes, individual (in)equality, the usage of property restriction are some of theses limitations. However, it provides with extensions the restricted subset of OWL DL by adoption the data type formalism of OWL-E(Pan et al., 2004), UNA, adding constraint constructs to *objectProperties*, Local CWA (LCWA is applied in order to enable constraint checking) and F-logic features for meta-modeling.

3.4 Summary : strength / weakness

Sections 3.1, 3.2 and 3.3 showed three main ontology models. All these models describe a domain in terms of classes, properties and individuals. Each of them adds to this kernel some extra features (constructors, axioms) enriching each system by specific useful features but none of them includes all the characteristics identified in section 2. All these characteristics are important in knowledge intensive applications, they should be provided by these knowledge models. Instead of creating a whole ontology model covering all the characteristics, which is an unfeasible task, we claim that each ontology model shall be used where it is sufficient.

- PLIB is most suitable for instance storage (canonical classes, strong property typing). It has an

elaborated data type system but do not allow reasoning. The ontoDB tested architecture compatible with other ontology models offers a persistent OBDB for PLIB ontologies.

- OWL DL offers a large number of constructors allowing reasoning tasks but cannot express constraints, the modeling context of concepts or measure values.
- OWL Flight combines the expressivity of OWL and query answering by extending ontologies with rules. It has a more elaborated data type formalism than OWD DL. It goes towards database modeling with LCWA, constraints and property typing but remains with non canonical concepts.

To conclude, we notice that while designing these ontology models a compromise between expressivity, instance storage and query answering has been made. Table 1 summarizes the comparison of PLIB, OWL DL and OWL Flight.

Table 1: Comparison of PLIB, OWL DL and OWL Flight.

	PLIB	OWL DL	OWL Flight
Strong typing	+ +	+ -	+ +
Constraints	+ +	+ -	+ +
WA	CWA	OWA	LCWA
Context	+ +	- -	+ -
Inheritance	simple	multiple	multiple
Reasoning	+ -	+ +	+ +

4 MAPPING RULES

We argue that in knowledge-based applications, all aspects are important : (1) the domain must be well-modeled; (2) individuals storage must be optimized according to existing database management systems; (3) the knowledge system must contribute to provide database consistency and to offer a powerful query answering.

As we have noticed in section 3.4, none of these models is complete regarding to the knowledge modeling characteristics of section 2 but it appears that they cover all these characteristics. One may want for example to switch from OWL to PLIB to get benefit of a better database performance, but this switching supposes to re-encode OWL ontologies into PLIB ones each time and requires to have a good knowledge of all ontology languages. Therefore to get benefit from the strength of the three models and existing tools, our proposal consists in defining a mapping between them. This mapping is either structural when we identify corresponding features in different models, or procedural when it requires a transformation procedure allowing to transform a given feature of the source model into another in the target model. Notice that we present four mappings OWL $\underset{PO}{\overset{OP}{\rightleftarrows}}$ PLIB $\underset{FP}{\overset{PF}{\rightleftarrows}}$ Flight allowing to go from one ontology model to another. We have chosen to use PLIB as a pivot model because it provides with efficient persistency model through the OntoDB Ontology Based DataBase (OBDB) model.

We have classified each transformation from a model (the source) to another model (the target) into one of the following three distinct groups :

- "*Common kernel*" : contains features axiomatized both in the source and in the target model. Given a source's feature of this group, there exists a combination of target's features which expressed its semantics. It is the case for example of class hierarchy and property's cardinalities.
- "*Encoding*" : is made of features which do not exist in the target model but which can only be represented by judiciously encoding some target's features.
- "*Not mapped*" : includes features which cannot be classified in the two precedent groups (i.e., they cannot be represented in the target model).

The remainder of this section presents mapping rules between the different ontology models.

4.1 Mapping OWL to PLIB

This section describes the mapping OP from OWL to PLIB. In the remainder of this paper, we manipulate PLIB and OWL through their abstract syntax. OP(Q) refers to the target PLIB entity on which the source OWL entity Q is mapped. Table 2 shows that both primitive and defined OWL classes are translated into PLIB *item_class*. The main difference between them is that only primitive concept extensions are explicitly represented in PLIB-OntoDB while a view is associated to each OWL defined class to compute their extension.

We have introduced an *item_class* PLIB resource LRC (Local Root Class), which is an *item_class* required each time an OWL ontology is mapped. This resource is used to constraint the type of OWL properties. As example, the scope (range resp.) of an OWL property which cannot be deduced using property axioms or logical characteristics is set to LRC. We are then ensured that there will be no semantic error even in case of imported ontologies. When two ontologies are imported, the scope of their properties is restricted by their respective LRC. Instead of reviewing

the whole OP mapping of table 2, let us review some elements for each group defined in section 4.

Table 2: Mapping OWL to PLIB.

OWL	PLIB
Class(A partial)	Item_class(A) is_a LRC
Class(B complete)	Item_class(B)
SubClass(A C_1) SubClass(A C_n)	OP(A) is_a OP(C_j), OP(A) is_case_of OP(C_i), i≠j, card(applic(OP(C_j))) ≥ card(applic(OP(C_i)))
SubClass(A Restriction(Q allValuesFrom(C)))	OP(A)(constraint(Q range(OP(Q)) = OP(C))), OP(A)is_a namescope(OP(Q)
comment	definition
ObjectProperty (P domain(A_1) range(A_2) [symmetric]	Non_dependent_p_det P(namescope(OP(A_1)) range(OP(A_2), Cards(0, ?))), OP(P) ∈ applic(OP(A_1)), note('NOTE&/symmetric/')
UnionOf(A B)	

Both OWL and PLIB offer support for simple inheritance, the corresponding mechanism is called either *subclass* in OWL or *is_a* in PLIB. However, multiple inheritance is not allowed in PLIB, but we have expressed it by combining the *is_a* and *is_case_of* PLIB relationships. Also, OWL property logical characteristics cannot be represented in PLIB, We propose to use the PLIB meta data *note* filled with a predefined formatted text (e.g., *note : = 'NOTE&/symmetric/'* for a symmetric property) to encode property's characteristics. Thus, although property characteristics are not axiomatized in PLIB, the predefined text encoded with the meta data *note* gives an essential information to understand their behavior. This information will be later used by OntoDB to achieve specific actions (e.g., saturate the database in case of symmetric properties). Some OWL constructs like *UnionOf(...)* cannot be mapped to PLIB. Notice that in OntoDB, OWL individuals are converted and represented when possible on their canonical form. For example an instance of the OWL class *Restriction (P ...)* will be stored like an instance of the PLIB class *OP(domain(P))*, while an instance of the OWL class *UnionOf(A,B)* will not be stored in OntoDB.

4.2 Mapping PLIB to OWL

When mapping PLIB to OWL, the main difficulty to avoid is the mapping of the rich PLIB data type system and context expression.

OWL data type formalism is not expressive enough to represent PLIB units. So they are translated to their basic type (see in table 3). We decided to associate the OWL meta data *comment* filled with a predefined text each time a PLIB data type is mapped. It will enable to further explain or clarify the PLIB data type (e.g., *comment := 'COMMENT&/unit&Symbol(u)/'* for a unit data type. Moreover, on the contrary of the mapping from OWL to PLIB where the translation of constructors was injective, the mapping from PLIB to OWL is not injective. The meta-descriptor *comment* will therefore be used to discriminate(make it injective). Notice that for example, PLIB context-dependent property cannot be mapped to OWL.

Table 3: Mapping PLIB to OWL.

PLIB	OWL
Item_class(A ...)	Class(A partial ...)
A is_case_of C_2	Subclass(PO(A) PO(C_2))
A(constraint(P range(P)= C))	Subclass(PO(A) Restriction(PO(P) allValuesFrom(PO(C)))
Non_dependent_P_D ET(P namescope(A) range(C Cards(n,?)))	ObjectProperty(P domain(PO(A)) range(PO(C))), Subclass(PO(A) Restriction(PO(P) minCardinality(n)))
Unit_type(T, u)	Integer(T comment('COM MENT&/unit&Symbol(u)/'))
definition(i)	comment('DEFINITION'i)
Dependent_P_DET(P)	

The detailed mapping from OWL to PLIB and from PLIB to OWL is available in(Fankam, 2006).

4.3 Mapping F-logic to PLIB

As noticed in section 3.3.4, OWL Flight is a restricted subset of OWL DL, so the translation rules from OWL DL to PLIB still holds with OWL Flight. Moreover, OWL Flight constraint axioms are translated in the same way as OWL DL restriction axioms.

Table 4 shows the correspondence allowing to map F-logic constructs to PLIB. Structural aspects for classes and properties are translated like for OWL. However F-Logic methods are translated into PLIB specific properties with aggregates in order to handle n-ary associations of properties to classes. We are currently investigating the mapping of other F-Logic specific features as rules.

4.4 Mapping PLIB to F-logic

The translation of PLIB to OWL still holds for OWL Flight for the same reasons previously mentioned. The most important change concerns PLIB properties cardinalities which are translated to OWL Flight

Table 4: Mapping Flogic to PLIB.

F-Logic	PLIB
A[...]	Item_class(A ...)
A :: C_1	FP(A) is_a FP(C_1)
A :: C_1 ... A :: C_n	FP(A) is_a FP(C_j), FP(A) is_case_of FP(C_i), i≠j, card(applic(FP(C_j))) ≥ card(applic(FP(C_i)))
A[Q =>> C]	Non_dependent_ P_DET(Q namescope(FP(A)) domain(FP(C) Cards(0, ?)))), FP(Q)∈ applic(FP(A))
Datatype(T)	Simple_type(T)
head ← body	

constraint axioms. In addition, some PLIB user defined data types and properties context definitions and properties values context are translated by OWL-E data type group. For example, the dependency (1) : *resistance = F(temperature)* and the derivation (2) : *temperature_in_celcsius = temperature_in_kelvin minus 273,15*, will be expressed in OWL-E as (1) *restriction(resistance, temperature allTuplesSatify(P1))* and (2) *restriction(temperature_in_Celcsius, temperature_in_kelvin allTuplesSatisfy(P2))* where P1 (P2 resp.) refers to the logical predicate corresponding to (1) ((2) resp.).

Table 5: Mapping PLIB to FLOGIC.

PLIB	F-logic
Item_class(A)	A
A is_a C_1	PF(A) :: PF(C_1)
A is_case_of C_2	PF(A) :: PF(C_2)
Non_dependent_P_DET(P namescope(A) domain(C))	PF(A)[PF(P) => PF(C)]
Simple_type(T)	Dataype(T)
Unit_type(T, u)	Dataype(T)
A(constraint(range(P)= C))	← ¬ ?o:PF(C) ∧ ?x:PF(A)[PF(P) => ?o]
Non_dependent_P_DET(P namescope(A) domain(C Cards(n,?)))	PF(A)[PF(Q)=>>PF(C)], ← ¬less(length(?o), n) ∧ ?x :FP(A)[FP(P) → ?o]
Dependent_P_DET(P)	

Table 5 shows the mapping between PLIB and F-logic. All PLIB constraints are mapped using F-logic rules. PLIB data types are translated to their corresponding basic type since F-logic only allows simple data type.

4.5 Applying MDE approach for implementation

Our goal is to allow a systematic transformation between models. By applying rules at the ontology model level, we are ensured that any ontology resulting from the mapping of a source ontology (seen as an instance of a source ontology model) preserves the properties of its data along the transformation and is valid according to the target model. The different mapping tables defined along this paper are data-oriented descriptions. It is possible to implement these mappings using model transformation technique(Bernstein, 2003). Indeed, transformations are implemented to map instances of data models encoding the meta model of the ontology models. The EXPRESS language and its transformation language EXPRESS-X is used for this purpose.

EXPRESS-X(ISO-10303-14, 1999) defines, manages, and maintains traces and relationships between different models. Like in other MDE techniques, the transformation between models is expressed in a mapping schema called *schema_map*. Transformations can then be written in the form of a generic program having as input the source model and the target model.

Table 6: An example of EXPRESS-X mapping.

SCHEMA source_S;	SCHEMA target_S;
ENTITY student; noSS : STRING; name : STRING; salary : optional REAL; END_ENTITY; END_SCHEMA;	ENTITY employee id : STRING; name : STRING; END_ENTITY; END_SCHEMA;

```
SCHEMA_MAP example_map;
TARGET target_S;
SOURCE source_S;
MAP st2emp AS emp :target_S.employee
FROM st : source_S.student;
WHERE (st.salary > = 0);
SELECT
  emp.name := st.name
  emp.id := st.noSS
END_MAP;
END_SCHEMA_MAP;
```

Table 6 shows an example of an EXPRESS-X mapping. It defines two schemas a source (*source_S*) and a target one (*target_S*) and a mapping schema (*example_map*). It maps *student* to *employee* by unfolding the properties of *student* into the target entity. Given the source instances of *source_S* :
#1 = student('601','Smith',$);
#2 = student('203','Jones',3500) .
The resulting target instances of *target_S* are:
#1 = employee('203','Jones').

The objective of this paper is to overcome the heterogeneity that results from different ontology models which were defined for different purposes. We have investigated the translation between ontology models, and more precisely the integration of other ontology models in the PLIB model and then in its OBDB OntoDB. We have first proposed six relevant characteristics(strong typing, constraint, WA, context modeling, inheritance, reasoning) which are very important to capture and to manage knowledge from the expressiveness and database points of view.

We have presented and compared with respect to these characteristics three ontology models PLIB, OWL DL and OWL Flight by emphasizing for each model on its main characteristics, its formal semantics, its modeling fundamentals and its usage. We summarized this comparison in table 1. With respect to this comparison, we concluded that none of these three ontology models is complete. We therefore argue, according to the fact that knowledge applications require and combine all the previously mentioned characteristics, that an automatic mapping and/or a procedural translation must be defined upon these models. By adopting PLIB as the central model for these mapping, we have provided an efficient management architecture compatible with other ontology models (OWL) in addition to the features which will enrich PLIB through the mappings.

We have identified a number of remaining important issues to extend this work, including: (1) Investigate the efficiency and the complexity aspects of this mapping. (2) Enrich the PLIB central model to formally represent non canonical expressions and features which are now translated using meta data. (3) Allow the definition of new meta classes in OntoDB in order to support specific features of other ontology models.

REFERENCES

Aklouf, Y., Pierra, G., Ait-Ameur, Y., and Drias, H. (2003). PLIB Ontology For B2B Electronic Commerce. pages 269–278. R. Jardim-GonÇalves, J. Cha and A. Steiger-GarÇao.

Baader, F., Calvanese, D., McGuinness, D., Nardi, D., and Patel-Schneider, P. (2003). *The description logic handbook*. Cambridge University Press.

Bechhofer, S., van Harmelen, F., Hendler, J., Horrocks, I., McGuinness, D., Patel-Schneider, P., and Stein, L. (2004). OWL Web Ontology Language Reference. *W3C, http://www.w3.org/TR/owl-ref/*.

Bernstein, P. (2003). Applying Model Management to Classical Meta Data Problems. In *Conf. on Innovative Database Research (CIDR'03)*.

de Bruijn, J., Polleres, A., Lara, R., and Fensel, D. (2004a). OWL-. *WSML Deliverable D20.1 v0.2, http://www.wsmo.org/2004/d20/d20.1/*.

de Bruijn, J., Polleres, A., Lara, R., and Fensel, D. (2004b). OWL DL vs OWL Flight : conceptual Modeling and Reasoning for the Semantic Web. Technical report.

de Bruijn, J., Polleres, A., Lara, R., and Fensel, D. (2004c). OWL Flight. *WSML Delivrable D20.3 v0.1, http://www.wsmo.org/TR/d20/d20.3/*.

Dehainsala, H., Pierra, G., and Bellatreche, L. (2007). An Ontology-Based Database for Data Intensive Applications. *To appear in : 12th Int. Conf. on Database Systems for Advanced Applications (DASFAA)*.

Fankam, C. (2006). Vers une intégration des différentes approches de modelisation base ontologique : application aux modèles PLIB et OWL. Technical report.

ISO-10303-14 (1999). Product Data Representation and Exchange - Part 14 :EXPRESS-X Language Reference Manual. Technical report, ISO.

ISO-13584-42 (1998). Industrial Automation Systems and Integration Parts LIBrary Part 42 : Description methodology : Methodology for Structuring Parts families. Technical report, ISO.

Kifer, M., Lausen, G., and Wu, J. (1995). Logical foundation of object-oriented and frame based language. *JACM*, 42(4):741–843.

Pan, J., Horrock, I., and Wu, J. (2004). OWL-E : extending OWL with expressive datatype expressions. *IMG technical Report*.

Patel-Schneider, P. and Horrocks, I. (2006). Position paper: A Comparison of two modelling Paradigms in the Semantic Web. pages 3–12.

Pierra, G. (2004). The PLIB ontology base approach to data integration. pages pp. 13–18. R. Jacquart, Kluwer Academic Publishers.

Pierra, G., Dehainsala, H., Ait-Ameur, Y., Bellatreche, L., Chochon, J., and Mimoune, M. E.-H. (2005). Bases de données a base ontologique. Principe et mise en oeuvre. *Ingénierie des Systèmes d'Information (ISI'05)*, 10(2):91–115.

Spiby, P. and Schenck, D. (1994). *The EXPRESS Language Reference Manual*. ISO-IS-10303-11. ISO Genève.

Xuan, D. N., Bellatreche, L., and Pierra, G. (2006). Ontology Evolution and Source Autonomy in Ontology-based Data Warehouses. *Revue des Nouvelles Technologies de l'Information (EDA'2006)*, pages 55–76.