Abstraction of Representation for Interoperation

David A. Maluf and Gio Wiederhold

Department of Computer Science
Stanford University
Stanford, CA 94305

Abstract. When combining data from distinct sources, there is a need to share meta-data and other knowledge about various source domains. Due to semantic inconsistencies, problems arise when combining knowledge across domains and the knowledge is simply merged. Also, knowledge that is irrelevant to the task of interoperation will be included, making the result unnecessarily complex. An *algebra* over ontologies has been proposed to support disciplined manipulation of domain knowledge resources. However, if one tries to interoperate directly with the knowledge bases, semantic problems arise due to heterogeneity of representations. This heterogeneity problem can be eliminated by using an *intermediate model* that controls the knowledge translation from a source knowledge base. The intermediate model we have developed is based on the concept of *abstract* knowledge representation and has two components: a modeling behavior which separates the knowledge from its implementation, and a performative behavior which establishes context abstraction rules over the knowledge.

1 Introduction

Many designers, developers, and users realize that information in private and public databases, as on the Internet, provides increasing opportunities to enhance productivity. Understanding the content of the available information requires the use of knowledge-based systems. However, effective use of knowledge to support problem solving also requires use of multiple knowledge sources. Simply taking the union of multiple knowledge sources derived from distinct domains creates several problems. One problem is due to the differing representations of knowledge obtained from different sources. Furthermore, the terms used to represent knowledge from diverse domains will have semantic inconsistencies. These inconsistencies occur because the knowledge-content will differ both in semantics and in compositional granularity. A union of multiple knowledge bases includes irrelevant knowledge and the result will be large, and disproportionally costly to process. For interoperation, the focus should be on the intersection of the knowledge, since intersection will define the required articulations. The term *articulation* refers to the rules that are used for knowledge which provides links across domains [6]. To solve these problems, many heuristic approaches have been proposed and implemented [21]. However, these approaches have limited applicability and make it difficult to establish and maintain large knowledge bases.

We extend and generalize the identification of the articulation to a set of manipulations, such as selecting, combining, extending, specializing, and modifying components from diverse common and domain-specific ontologies. To deal with most of these issues, an algebra over ontologies has been proposed in [26] which is intended to support disciplined manipulation of knowledge resources. The representation of vocabularies and their structure is termed an *ontology* whereas the operations that combine and partition structures in a sound and well-behaved manner are termed an *ontology algebra*. The basic algebra consists of three operations, namely intersection, union and difference.

The objective of an ontology algebra is to provide the capability for interrogating many knowledge resources, which are largely semantically disjoint, but where *articulations* have been established that enable knowledge interoperability. The emergent need to define articulations between knowledge resources has been demonstrated and described in [15][27].

Although this paper does not describe the ontology algebra itself, it is motivated by it and presents two aspects, namely converting knowledge representations and partitioning.

Representation: When designing an ontology algebra for diverse ontologies and knowledge-based systems, it is important to understand the constraints that an underlying knowledge representation poses on the knowledge content. By overcoming these constraints, one can expect an increase in knowledge-sharing capabilities [21]. An early example of this approach revolves around porting ontologies from one knowledge representation language into multiple ones as done by *Ontolingua* [14]. Ontolingua is a mechanism for translating from a standard syntax into multiple-representation systems. However, directly translating entire ontologies to multiple representations leads to irrelevant knowledge, semantic inconsistencies, and disproportionally large knowledge-bases. On the other hand, imposing the ontology algebra as part of the multiple representation systems is not feasible. This can be simply demonstrated by observing that most industry standards only support declarative interfaces such as the CORBA Interface Repository [23]. In other words, the algebra requires its own workspace. Our hypothesis is that an ontology algebra to combine and partition knowledge is feasible when provided with an *intermediate* model. The intermediate model is declarative following modern concepts [20]. It establishes a rule-based environment to sustain operations envisaged by the ontology algebra.

Partitioning: In this paper, we also addresse the problem of how to abstract and entail encoded knowledge within contexts. We will formulate the foundation of knowledge abstraction as a basic problem in propositional calculus. Knowledge abstraction as used in this paper composes declarative ontological compositions, keeping their context through formal predication. These transformations will establish the *articulation axioms* for the ontology algebra. The articulation axioms represent the partitioning of a knowledge model and are maintained within an intermediate model.

The intermediate model produces the environment needed to provide users and system developers with the ability to manipulate knowledge bases and domain-specific ontologies. These manipulations will support the interoperation of descriptions of topics of interest when using the knowledge base. These descriptions are reusable by multiple applications that need to access to diverse knowledge and data sources. The descriptive formalism makes the intermediate model maintainable in rapidly changing environments.

2 Knowledge Representations and Interoperation

The development of an *intermediate* model reported in this paper is motivated by the interoperability among existing knowledge-representation formalisms. The series of knowledge representation formalisms and frameworks starting with KL-One [4] and currently culminating in systems like Classic [3] and LOOM [19] provide powerful tools and knowledge expressiveness. However, they were not intended to interoperate. How much has to be added in their infrastructure and reasoning capability to achieve knowledge interoperability is still unclear. There have been two recent efforts that open up possibilities for meaningful knowledge interoperation: the development of context logic [18] and knowledge interfaces for sharing [21]. The advance in context logic is the notion of translating encoded knowledge relative to its context. This is the approach taken in the reengineering of Cyc [17] where micro-theories confine the contextual differences [15]. Advances in knowledge sharing revolve around translating knowledge bases from one representation formalism to multiple ones. However, the problem of translating knowledge bases across different representations is difficult to implement when translation is to occur in all directions.

Most knowledge representation formalisms are bound to specialized methods of inference but a few have formats that focused on reuse rather than inferencing [16][13]. These formats provide a common denominator and support a solution for interoperability. However, manipulation of source context preserves deeper knowledge, and with the planned ontology algebra, should bring about better knowledge scalability. Abstraction in context is essential since different ontology compositions have different context granularities and hence cannot interoperate directly.

In this work we support knowledge abstraction between different knowledge compositions. We define performative rules to maintain the abstraction as part of the knowledge partitioning. The process of abstraction emphasizes the importance of separating knowledge from its implementation. The notion of separation was initially suggested in the scheme of the Agent Communication Language [12]. We realize that there is much leverage in overlaying the knowledge sources with the needed partitioning because that is where the context is best understood.

2.1 The Intermediate Knowledge Model

The intermediate model scales and partitions knowledge bases and domain ontologies given some application objectives. An intermediate model presents two facets that can operate concurrently or independently, namely a *modeling* behavior and a *performative* behavior. The modeling behavior translates the knowledge into axioms and prescribes the notion of *Articulation Axioms*. The performatives maintain the articulation axioms within contexts where *Articulation Rules* can be established. These articulation rules are used for linking knowledge across domains. To permit information from distinct sources to be accessed and operated upon, a consensus should be reached on the rules to formulate the articulation axioms. In our approach these rules are declared independently from the domain translation intermediate models. Figure 1 illustrates the two functionalities of an intermediate model: (i), the translation of knowledge from a knowledge base or knowledge repository (models A and B) to its corresponding intermediate model, and (ii), the articulation rules which are maintained through a separate intermediate model (Model C).

Most knowledge-based systems maintain knowledge models by interacting the knowledge acquisition, knowledge representation, and inference components. Similar knowledge models have a tendency to be formal in their design and involve much implicit knowledge. Correspondingly, we believe that by restricting these representations to a declarative form, we can achieve less formality, thus more abstraction. To this end, the intermediate model considers a non-formal knowledge modeling approach which has been adapted by many KBMS developments [14][29], but that we keep within a propositional calculus-capable *interlingua*. The implementation of the intermediate model employs the first-order logic Knowledge Interchange Format (KIF) [11] as its interlingua and postulates the hypothesis that the composed knowledge is independent of its representation.

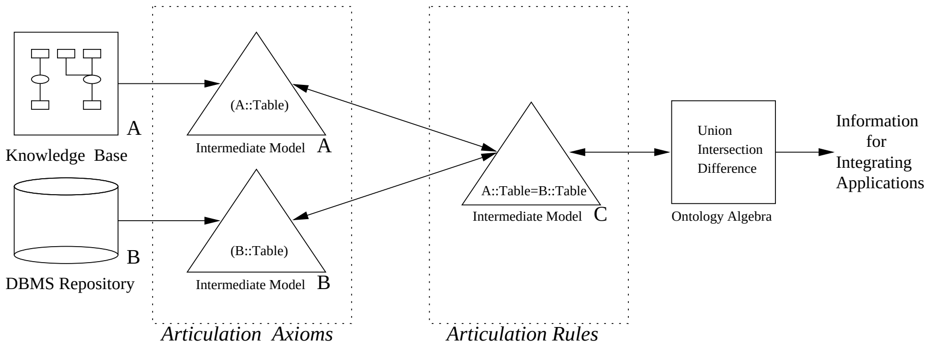

Fig. 1. When interoperation among multiple knowledge-based systems, Articulation axioms establish the partitioning of knowledge. On the other hand, articulation rules establish the links between domains and drive an ontology algebra.

Vocabulary and Relations The translation operates on a vocabulary which describes the constituents of the ontologies and contains the ontological commitments that are required for the interoperation. The vocabulary is explicitly required, since the vocabulary defines the domain knowledge. By "vocabulary" in this paper we mean an expression or a term that belongs to a dictionary. Translating the vocabulary independently of the specific base information, i.e. the current relationships among the constants, assures complete semantic coverage of the vocabulary. The vocabulary alone has no structure and is only defined by the names. This temporary decomposition of an ontology enables the agglomeration of the terms into one set of common vocabularies. A property in the implementation of the intermediate model is it performs a unification on the terms involved. "Unification" in this paper is used broadly to handle simple ontological composition details such as grammar, spelling, and word composition.

The relationships among the constants refer to the relations among the vocabulary. The relation constants establish the ground or atomic meanings within a knowledge base and apply to the predicates that maintain the relationships between the vocabulary [9].

2.2 Terminology, Definitions and Assumptions

Axioms: Predicates are the basic construct of declarative knowledge. For example, one can express the fact that a `Book` is above the `Table` by taking a relation symbol such as `Above` and defining a predicate `Above(x,y)`. Hence, for the object symbols `Book` and `Table` we can declare the proposition `Above(Book, Table)` [10]. Often, a predicate contains semantic conjunctions or disjunctions within its syntax [7] to express complex relation constructs.

To deal with the inadequacy of semantics and to conform to the syntactics of predicate logic, structured predicates are separated into simpler atomic propositions. If for example a knowledge base considers the proposition `Above(Book` *and* `Pen,Table)`, the intermediate model finds it is equivalent to the conjunction of the following two predicates `Above(Book, Table)` $\wedge$ `Above(Pen, Table)`. In general, predicates are atomic and do not contain semantic operators. □

Abstraction: Abstraction is equivalent to the production of simpler approximations of domain knowledge bases driven by approximation rules. When knowledge bases involve a large vocabulary, abstraction is also the process of aggregating the knowledge model to another involving smaller vocabulary and fewer constants. Often the aggregation is performed by translating the declarative knowledge predicates and grouping the vocabulary and constants into arbitrary well-formed formulas. In [1], one can distinguish the different types of abstraction such as qualitative abstraction, quantitative abstraction, terminological abstraction and temporal abstraction. In our work, however, the notion of abstraction focuses on manipulating knowledge within context.

The principal idea in abstraction is that a knowledge base includes a number of levels of abstractions. For example, in the case of applying the relation symbol `Above` to the objects `Book` and `Table` and a third argument denoting a situation s, say {`Library, Office, Home`}. Abstraction in granularity is achieved when the proposition `Above(Book,Table,`s`)` is translated into `Above(Book,Table)`. □

Context: Context has been proposed as a means of defining the validity of a sentence relative to a situation. Formalizing contexts [18][15] develops the notion of context which allows predicate axioms for fixed situations to be "lifted" to more dynamic contexts where situations change. The context formalism is an extension to first-order logic in which sentences are valid within a context. To this end, we use the denotation of $\mathtt{ist}(c;p)$ such that we have a formula of a proposition p which is true in a context c. For example given a context `Office`, one can write `ist(Office; Above(Book, Table))`. In this paper we drop `ist` and consider a concise and simple form as in the formulas `(c;p)`. From the previous example we may write the proposition `(Office; Above(Book,Table))`. In the implementation of the latter example, the intermediate model assumes a template pattern of the form `(AXIOM:00123 (CONTEXT Office) (RELATION Above) (OBJECT Book Table))`.

At this point it is worth noting a difference between the context logic formalism approach and the one in this paper which is that context logic defines a default coreference rule which states, that as a default, the meaning of a symbol does not change from one context to another. We consider that symbols never mean the same. The key in resolving ambiguity of meanings is in establishing and manipulating contexts. Formally we consider for every proposition p referring to a pair of objects x and y, then x is asserted as a possible context of the proposition p (*Modus ponens*) or

$$\forall x : \quad p(x,y) \Rightarrow (x;p(x,y)).$$

For instance, as one may consider the proposition `isa(Furniture,Table)`, the proposition may be reformulated and asserted as `(Furniture;isa(x,Table))` which reads `isa(x,Table)` in the context of `Furniture`. Note at this point that we assume there is no need for a proposition to refer to the object as an argument. The use of the variable `x` is to maintain the correct arity of the predicate `isa`. Similarly, the proposition `isa(Database,Table)` may be reformulated and asserted as `(Database;isa(x,Table))`. Hence `isa(x,Table)` has two contexts, namely `Furniture` and `Database`. □

Articulation Axioms: The idea of using context directly relates to the notion of abstraction in this paper. However manipulating context as an abstraction process presents another scope, namely *Articulation.* Articulations have two facets and are identified separately as *articulation axioms (AA)* and *articulation rules (AR)*. The articulation axioms are the axioms upon which these rules operate. In simple words, articulation axioms are the partitions which can be matched from one domain to another. The axioms are formulated within a domain or knowledge model whereas the rules are maintained separately from the domain [27]. □

3 Knowledge, Formulation and Rewriting

The previous sections defined a few requirements of an intermediate model. However, these requirements place no restrictions on the possible context entailment propositions can have. In this section, we explore the effect of the intermediate model first on the properties to model semantic distinctions with context.

In this section we also explore the implication of the intermediate model on translating database domain knowledge. We assume that the underlying knowledge has been translated from declarative languages which correctly conceptualized the domain knowledge as propositions. The intermediate model addresses several tasks in translating database domain knowledge, namely resolving implementation differences, interpretation and partial information.

1. *Resolving implementation differences:* As some of the work on designing databases focuses on designing data models, one can realize the different possibilities in the conceptual modeling considered in their design. For an interoperability problem such as in data integration process, one should focus on relating different data models, e.g., mapping the relational model to the object model which requires structural knowledge [25]. However, even if we consider only databases using the same data model, there are significant differences which make the task of relating the semantics of the data model difficult. These differences are due to their schema composition.
2. *Interpretation:* To permit the explicit knowledge as in the case of databases to interoperate with other sources, it is not sufficient to simply merge the information on the basis of the vocabulary. Simply matching vocabulary does not correspond to matching meanings. On the other hand, considering the ontologies of these domains would be a better tradeoff in the interoperation. For each of these domain schemas, their corresponding ontology is examined in parsing their vocabulary and the specification of their relationships. Interoperability can occur in a sound manner with propositions.
3. *Partial information:* Declarative knowledge can deal with handling incomplete information [10]. This problem in database interoperability is simply typified by the symptoms of most directed graphs which is their inability to handle partial information. For example there is no way to assert a proposition in a object hierarchy without a reference to a root object. The lack of reference in general is often found with systems that lack external schemas. Similar partial information populates most semistructured information systems, e.g., the World Wide Web.

3.1 Example: Interpreting Primitive Models

Primitive models are models that are specialized in their design. For example, one can consider a two-column tabulation as a specialized method in representing data in binary predicates. Another primitive model is the Entity Relationship (ER) model which is used in the conceptual phase of database design. Most primitive models are graphical languages although they have been expanded using an extension of the relational algebra [8][28] We will discuss their strengths and weaknesses in our setting.

We focus on the common construct in these specialized language, namely relationships. Various relationships may hold within primitive models, as taxonomic relationships, arbitrary relationships, hierarchical, cyclic, acyclic. However, the basic construct of relationships are their directed graph representation, namely labeled arcs and nodes.

One can interpret the directed graph by postulating each labeled arc with the corresponding objects into axioms. The term *interpretation* admits a formal definition in declarative knowledge and relates to a mapping process. For instance, in an object taxonomy, one may write `isa(Object,Table)` where `Table` is a subclass of `Object`. In a semantic network, one may write `Above(Book, Table)` where an arc `Above` relate `Book` to `Table`, etc. Formally, the declarative interpretation is taken as the implication of a performative pattern where a binary but simple proposition is formed from a relation r and terms x and y by combining them as $\mathtt{r}(\mathtt{x}, \mathtt{y})$. Figure `2.a` illustrates a binary segment of a directed graph.

Context and Axioms Declarative knowledge is an excellent interlingua, however it does have its share of restrictions. In fact much of the work in declarative knowledge revolves around an assumption of one-term-one-meaning mapping. Our position is that we consider that symbols never mean the same in different contexts. This sets our approach apart from predicate logic [10] and context logic [15][5], We have two main reasons. First, when considering knowledge that has been composed using standard knowledge acquisition and concept modeling tools, one cannot expect that terms refer to the same context. For instance, `Drug(Marijuana)` can be administered in different contexts, namely `Recreational` and `Medical`. Secondly, when knowledge has been formulated simply as a union of multiple domains will result in a knowledge model where terms have multiple contexts and misinterpretations is likely.

Let us assume an example when a matching has occurred across domains. We consider the matching terms from two domains where specifically we focus on the resulting graph segments. The segments are "Table subclass-of Database", "Table subclass-of Furniture", "Furniture subclass-of Object" and finally "Table subclass-of Object". We formulate the corresponding binary interpretations which result in `isa(Database, Table)`, `isa(Furniture, Table)`, `isa(Object, Furniture)`, and `isa(Object, Table)`. Their corresponding implementation is shown in the following table.

VOCABULARY	RELATION CONSTANT	BINARY EXPRESSION
(OBJECT Table)	(RELATION isa)	(AXIOM:001 (nil) isa (Object Table))
(OBJECT Database)		(AXIOM:002 (nil) isa (Furniture Table))
(OBJECT Furniture)		(AXIOM:003 (nil) isa (Database Table))
(OBJECT Object)		(AXIOM:004 (nil) isa (Object Furniture))

While we can identify the vocabulary (column 1) from the relations (column 2), we can assert a set of propositions (Column 3) using the interpretation of binary relations. What is of interest in this example is that within the same model the object `Table` is asserted within two independent contexts, namely `Database` and `Furniture`. In formulating an interpretation as a binary relation, we have implicitly assumed a conjunction between the propositions in the intermediate model. However the truth of the implicit assumption is not correct. For instance, considering `isa(Furniture, Table)` $\wedge$ `isa(Database, Table)` is syntactically but not semantically correct. To assert `isa(Furniture, Table)` and `isa(Database, Table)` concurrently we should consider a disjunction `isa(Furniture, Table)` $\vee$ `isa(Database, Table)`. On the other hand, a conjunction is semantically correct between `isa(Object,Table)`$\wedge$`isa(Object, Furniture)`. The difficulty in interpreting primitive models in declarative knowledge propositions, is the inability to manage the assertion of a symbol concurrently in different contexts. This simply reflects the default coreference rule which states that as default the meaning of a symbol does not change from one context to another [15]. The coreference rule is not valid when dealing with multiple domains.

Rewriting Binary Axioms The problem encountered in the previous section is not as serious as it appears. The solution is simple and lies in investigating the meaning of the directions in the graph which are considered as the argument index in the proposition.

To remedy the problem, we interpret the meaning of directions as the inheritance of *context* as for instance `Table` has `Database` as context in `isa(Database, Table)`. Once the contexts have been established on the previous example is applied, we realize that all implicit conjunctions semantically hold. The interpretation above is rewritten and asserted as `(Database; isa(x, Table))` and reads "Table subclass-of Database in context Database".

Rewriting *N*-ary Axioms The approach taken in Section 3.1, also taken by the database community, provides interpretation that formulates only binary relations. However, binary interpretations do no expose the full meaning of the terms used in the propositions. The binary interpretation of `isa(Object, Table)` does not characterize its meaning. If we reconsider the example stated in Section 3.1, `isa(Object, Table)` is true in two domain-independent contexts: `Furniture` and `Database`. `Table` unveils its context only when stated with an additional interpretation that contains `Table`, namely `isa(Furniture, Table)` or `isa(Database, Table)`.

In our approach, we generalize the problem and define the class of N-ary expression to include an ordering of N binary propositions. To this end, the ordering of binary expressions is performed in two possible directions, namely *spanning*, and *specializing*. *Spanning context* is the formulation of the axioms which depict the contexts a proposition has. Schematically, one can illustrate spanning context as multiple inheritance in directed graphs (Figure 2.`b`). On the other hand, *specializing* is the formulation of the axioms which depict for a context the propositions it relates to. For example, given the context `Object` for the proposition `(Object; isa(x, Furniture))`, one can specialize the context `Object` with `(Object; isa(x, Table))`. Schematically, one can illustrate a specialization as branching in directed graphs (Figure 2.c).

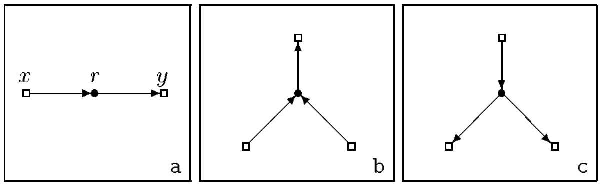

Fig. 2. Directed Graphs: Case a Shows a binary relation between two objects. Case b shows an N-ary relation where multiple objects refer to one object. Case c shows an alternative N-ary relation where one object refers to multiple ones. Note that a relation r is equivalent to the concept of arcs and in this paper this model is rather a generalization of directed graphs.

In principle, N-ary expressions are expressions which are not by default binary enforced by certain specialized primitive models. We consider the merging of binary expressions into N-ary expressions over the contexts. Hence, an N-ary expression is the logical attachment of N binary expressions where $N > 1$. For instance we may attach `(Furniture; isa(Furniture, Table))` and `(Object; isa(Object, Table))` into a single axiom or `(Object, Furniture; isa(x, Table))` which reads that `Table` has two contexts: `Object` and `Furniture`. In the implementation of the intermediate model, N-ary expressions are composed by rules. The rules that formulates these N-ary expressions must be maintained separately from the domain knowledge. Illustrated graphically as in Figure 2, N-ary expressions are constructs and contain much knowledge about where different tracings and connectivities diverge. The covariancy is the N-ary set serving as articulation axioms to a contravariant set of binary expressions. This criterion strengthens the meaning of partitioning with N-ary expressions. Hence the output of implementation of N-ary expressions of the example stated in Section 3.1 extends to:

```
NARY EXPRESSION OF TYPE SPANNING

(AXIOM:005 (Object Database) isa (Table))
(AXIOM:006 (Object Furniture) isa (Table))
```

```
NARY EXPRESSION OF TYPE SPECIALIZING

(AXIOM:007 (Object) isa (Table Furniture))
```

For the added values considered by being dynamically formulated, N-ary expressions serve numerous basic roles in knowledge rewriting, as they can provide simple articulation axioms and yet afford the partitioning of a knowledge model.

3.2 Axiom Formulation and Rewriting

Our interest is in expressing axioms in the framework of propositional calculus. This interest is based on a combination of two features of the intermediate model.

First the knowledge needed can be expressed in a form more or less independent of the uses to which the knowledge might be part of. Second the reasoning performed by the partioning process involves basic but simple logical operations on these propositions.

In the implementation of the intermediate model, we specify the proposition rewriting within first-order logic and use the formalism to constrain the propositions in terms of their context. A context can be thought of as a set of terms labeling a set of propositions. Intuitively, we assume a context production rule which states that the meaning of a proposition admits the context defined by the symbols stated within the proposition. For example, the proposition `Above(Book, Table)` has as possible contexts `Book` and `Table`. Although the definition of the context production rule is not very suggestive, it is not the case when considered within the framework of propositional calculus and context logic. In general, we consider the formulas as propositions of the form

$$(c_1, ..., c_M; p_1, ..., p_N) \tag{1}$$

which are to be taken that the propositions $p_1, ..., p_N$ are true in the contexts $c_1, ..., c_M$. For example, if we consider the proposition `(Office, Book; Above(x, Table))`, then we know that predicates `Above(x, Table)` is true in the context of `Office` and `Book`. The aim of reformulation context is not to use deduction as the computational framework, but rather to integrate axioms into optimal articulation axioms when the interoperation objectives are clear.

One can get concerned with the amount of possible propositions that can be calculated from Equation 1. We simplify the problem of focusing only on the articulation needed for interoperation. Since automated inferences are potentially capable of processing the symbolic propositions, the need for rules about how to process the axioms becomes essential. Although there are no general rules in establishing the rewriting of the propositions, the intermediate model supports the two performative rules: *spanning context* and *specializing*. Both reduce the scope of the interoperation.

1. *Spanning Context:* by providing a proposition with context such as considering the conjunction of the proposition `(Database; isa(x, Table))` to `(Object; isa(x, Table))` from the example stated in Section 3.1 and having `(Database, Object; isa(x, Table))`. Formally we have
$(\mathtt{c}_1;\mathtt{p}) \wedge (\mathtt{c}_2;\mathtt{p}) \Leftrightarrow (\mathtt{c}_1,\mathtt{c}_2;\mathtt{p})$
2. *Specializing:* by providing a context with propositions such as providing the proposition `(Object; isa(x, Furniture))` to the proposition `(Object; isa(x,Table))` and having `(Object; isa(x,Furniture), isa(x,Table))`. Formally we have
$(\mathtt{c};\mathtt{p}_1) \wedge (\mathtt{c};\mathtt{p}_2) \Leftrightarrow (\mathtt{c};\mathtt{p}_1,\mathtt{p}_2)$

Since we deal with propositions, the rules of first order and context logic apply. When the number of propositions is zero ($N = 0$ in Equation 1), then the vocabulary has its own context. For instance we have the list {`Office, Table, Book`}.

Another important possibility when rewriting the axioms is that propositions are always asserted within other axioms in a recursive form. Henceforth given the general denotation of Equation 1, we have recursively $(\mathtt{c_i};(\mathtt{c_j};.,.),.)$ and subsequently `(Object; (Furniture; isa(x,Table)))`.

In general, the achievement in recursively rewriting the context and propositions deals directly with the critical and difficult step in context abstraction and is also a contribution of this paper. Although the problem of interoperating with recursive definitions is difficult to achieve with minimal inferencing, rewriting the context recursively has two advantages. (i) it maintains the connectivity of the knowledge and (ii) it provides one way to control the context abstraction. The latter is achieved by asserting one context for each axiom. The current implementation does not support recursive definition.

Another potential interest in recursive rewriting is that it converges to the Object Extended Model (OEM) formalism which has been widely used, namely as the interlingua for The Stanford-IBM Manager of Multiple Information Sources (TSIMMIS) [30]. OEM is a self describing object model with nested identity. Every object in OEM consist of an *identifier* and a *value*. The value is either atomic, or set of objects, denoted as set of $\{$*label, id, value*$\}$. We refer to the *label* and *value* as context and axioms respectively.

It should be noted that one of the innovations of the intermediate model is that the proposed articulation axioms need not be static. The partitioning of the domain knowledge is dynamic where articulation axioms are asserted and retracted independently of the underlying knowledge base.

3.3 Status

The intermediate model is currently written in the 'C' Language Integrated Production System 6.0 (CLIPS) [22], a widely-available and easily portable expert system shell. Since user interface functions and data access functions are separated out into other components, the intermediate model consist mainly of rules. The wrapper that translates KIF to CLIPS facts is based on the standard KIF 'C' parser developed at Stanford University (http://logic.stanford.edu/software/kif) [11].

Figure 3 illustrates the current state of the interface of the intermediate model based on Hardy [24]. Hardy is a programmable diagramming tool. We use Hardy to assist the intermediate model in managing knowledge across domains. For the HP-Product domain (white background), a pattern matcher proposed three axioms which found in the Computer-Device domain (white background). The HP-Product and the Computer-Device domain ontologies are available from the Ontolingua server (http://www-ksl-SVC.stanford.edu:5915).

4 Conclusion

This paper presents an approach that uses context formalism in the development of standard knowledge representations and knowledge sharing and plays a role in knowledge interoperability. The context approach provides a powerful tool to define the validity of knowledge relative to a situation. This paper address the problem of how to abstract and entail encoded knowledge within contexts.

We describe an environment to interface underlying knowledge resources to the outside world. The objectives set in this paper are to establish the intermediate model needed to sustain knowledge interoperability and to produce the needed environment. Hence, users and system developers can translate knowledge bases that provide comprehensive but simple coverage of topics of interest, knowledge usability and reusability by different applications and knowledge maintenance in rapidly changing environments. The intermediate model can bring about a shift from designing knowledge base to the manipulation, enhancement, and maintenance of domain ontologies. The main objective of the intermediate model will be to handle an ontology algebra that combines and partitions structures in a sound and well-behaved manner.

The current research is a complementary approach to the current knowledge-based systems that support disciplined manipulation of knowledge resources.

5 Acknowledgments

The authors would like to thanks Oscar Firschein who carefully reviewed the draft of the document and contributed to its revised content. Input from John McCarthy at Stanford University has been particularly helpful.

References

1. M. Aben, "KADS-II CommonKADS Inferences"; Technical Report M2-UVA-041-1.0, University of Amsterdam, 1993.
2. A. Borgida, "From type systems to knowledge representation: natural semantics specification for describing logics"; Intelligent and Cooperative Information Systems, 1(1):93-126, 1992.
3. A. Borgida, R. Brachman, D. McGuinness, L. Resnick, "CLASSIC: A Structural Data Model for Objects"; SIGMOD Conference pages 58-67, 1989.
4. R. Brachman and J. Schmolze, "An Overview of the KL-ONE knowledge Representasion System"; Cognitive Science, 9(2):171-216, 1985.
5. S. Buvac and R. Fikes, "A Declarative Formalization of Knowledge Translation"; Knowledge Systems Laboratory, KSL-94-59, August 1994.
6. C. Collet, M. Huhns and W. Shen, "Resource Integration Using a Large Knowledge Base in Carnot"; IEEE Computer, 12(24), Dec. 1991.
7. K. Chang, H. Garca-Molina and A. Paepcke, "Boolean Query Mapping Across Heterogeneous Information Sources"; IEEE Transactions on Knowledge and Data Engineering, 8(4):515-521, Aug, 1996.
8. R. El-Masri and G. Wiederhold: "Data Model Integration Using the Structural Model"; Proceedings 1979 ACM SIGMOD Conference, pages 191-202.
9. J. L. Garfield (editor): "Modularity in Knowledge Representation and Natural Language Understanding"; MIT Press, 1987.
10. M. Genesereth and N. Nilsson, "Logical Foundations of Artificial Intelligence"; Morgan Kaufmann Publishers, Inc., Los Altos, CA, 1987.

11. M. Genesereth, "Knowledge Interchange Format"; Technical Report Logic-92-1, Stanford University, 1992.
12. M. Genesereth, N. Singh and M. Syed, "A Distributed and Anonymous Knowledge Sharing Approach to Software Interoperation"; Third International Conference on Information and Knowledge Management, 1994.
13. M. Gogolla, R. Herzig, S. Conrad, "Integrating the ER approach in an OO environment"; In Entity-relationship approach: Lecture notes in computer science. Springer-Verlag, Berlin; New York, 1994.
14. T. R. Gruber, "Ontolingua: A mechanism to support portable ontologies"; Technical Report, KSL-91-66, Knowledge System Laboratory, Stanford University, Stanford, CA.
15. R. V. Guha. "Context: A Formalization and Some applications"; Doctoral Dissertation, Stanford University, 1991.
16. B. L. Humphreys and D.A.B. Lindberg, "The Unified Medical Language Project: A Distributed Experiment in Improving Access to Biomedical Information"; MEDINFO 92, North-Holland, 1992, pages 1496-1500.
17. D. Lenat and R. Guha, "The Evolution of CycL, The Cyc Representation language"; Special Issue on Implemented Knowledge Representation System, Sigart, ACM, 2(3), pages 84-87, June 1991.
18. J. McCarthy, "Notes on Formalizing Context"; In proceedings of the Thirteenth International Joint Conference on Artificial Intelligence, 1993.
19. R. MacGregor and R. Bates, "The LOOM Knowledge Representation Language"; in Proceedings of the Knowledge-Based Systems Workshop, April 1987.
20. N. Singh, and M. Gisi, "Coordinating Distributed Objects with Declarative Interfaces"; in Coordination Languages and Models, Lecture Notes in Computer Science, 1061, Springer, 1996, pages 368-385.
21. R. Neches, R. Fikes, T. Finin, T. Gruber, R. Patill, T. Senator, and W. R. Swartout, "Enabling technology for knowledge sharing"; AI Magazine, 12(3):36-55, 1991.
22. G. Riley, "CLIPS: An Expert System Building Tool"; Proceedings of the Technology 2001 Conference, San Jose, CA, December 1991.
23. Sciore, Edward, Siegel, Michael, Rosenthal, Arnon, "Using semantic values to facilitate interoperability among heterogeneous information systems"; ACM Transactions on Database Systems, Vol.19, No. 2 pages 254-290, 1994.
24. J. Smart and R. Rae, "Hardy"; Technical Report Hardy User Guide, Artificial Intelligence Applications Institute, University of Edinburgh, 1995.
25. G. Wiederhold, T. Barsalou, B. S. Lee, N. Siambela and W. Sujansky, "Use of Relational Storage and a Semantic Model to Generate Objects: The PENGUIN Project"; Database '91: Merging Policy, Standards and Technology, The Armed Forces Communications and Electronics Association, Fairfax VA, June 1991, pages 503-515.
26. G. Wiederhold, "An Algebra for Ontology Composition"; Proceedings of 1994 Monterey Workshop on Formal Methods, U.S. Naval Postgraduate School, Monterey CA, pages 56-61, 1994.
27. G. Wiederhold, "Interoperation, Mediation, and Ontologies"; International Symposium on Fifth Generation Computer Systems (FGCSOB94), Workshop on Heterogeneous Cooperative Knowledge-Bases, Vol 3, pages 33-48, ICOT, Tokyo, Japan, 1994.
28. G. Wiederhold X. Qian, "Modeling Asynchrony in Distributed Databasesd"; Third IEEE Computer Society Data Engineering Conference, Los Angeles, Feb. 1987.

29. B. Wielinga, Walter, G.Schreiber, and H. Akkermans, "KADS-II expertise model definition document."; Technical Report M2-UVA-026-1.1, University of Amsterdam, 1993.

30. Y. Papakonstantinou, H. Garcia-Molina and J. Widom, "Object Exchange Across Heterogeneous Information Sources"; International Conference on Data Engineering, 1995.

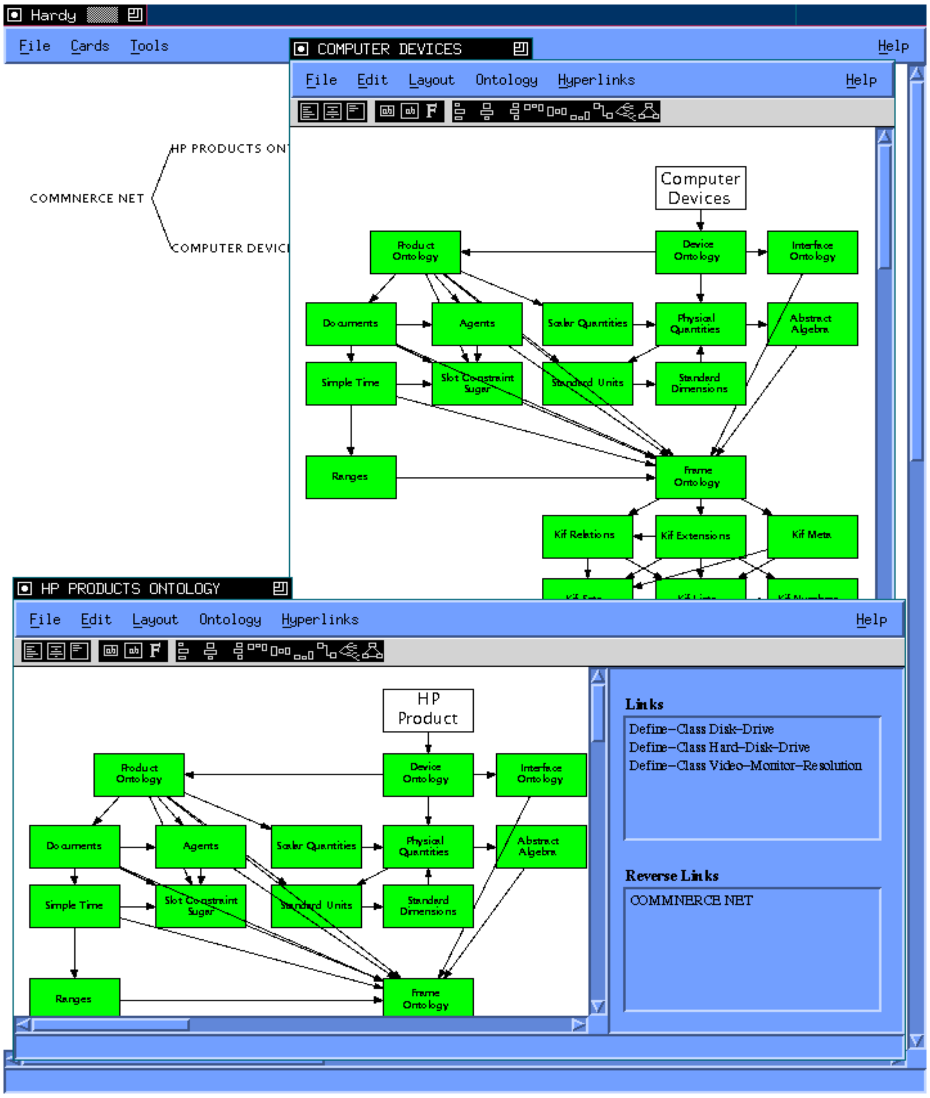

Fig. 3. The implementation of the intermediate model merged with Hardy, a programmable diagramming tool.