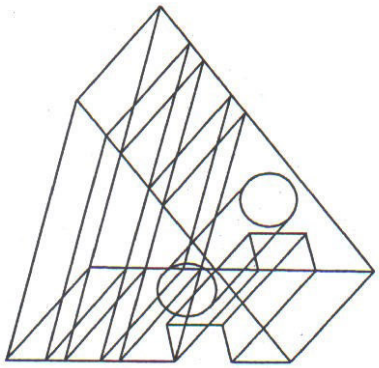


Contexts and Ontologies: Theory, Practice and Applications

Papers from the AAAI Workshop

Technical Report WS-05-01

AAAI Press

American Association for Artificial Intelligence

Contexts and Ontologies: Theory, Practice and Applications

Papers from the AAAI Workshop

Pavel Shvaiko, Chair
Jerome Euzenat, Alain Leger,
Deborah L. McGuinness,
and Holger Wache, Cochairs

Technical Report WS-05-01

AAAI Press
Menlo Park, California

The American Association for Artificial Intelligence
445 Burgess Drive
Menlo Park, California 94025 USA

ISBN 1-57735-237-8 WS-05-01

Manufactured in the United States of America

Organizing Committee

Jerome Euzenat (Cochair)
INRIA Rhone-Alpes, France
e-mail: Jerome.Euzenat@inrialpes.fr

Alain Leger (Cochair)
France Telecom R&D Rennes, France
e-mail: alain.leger@rd.francetelecom.com

Deborah L. McGuinness (Cochair)
Stanford University, USA
e-mail: dlm@ksl.stanford.edu

Pavel Shvaiko (Chair)
University of Trento, Italy
e-mail: pavel@dit.unitn.it

Holger Wache (Cochair)
Vrije Universiteit Amsterdam, Netherlands
e-mail: holger@cs.vu.nl

This AAAI–05 Workshop was held July 9, 2005,
in Pittsburgh, Pennsylvania USA

Program Committee

Alex Borgida, Rutgers University, USA
Paolo Bouquet, University of Trento, Italy
AnHai Doan, University of Illinois, USA
Jerome Euzenat, INRIA Rhone-Alpes, France
Richard Fikes, Stanford University, USA
Timothy W. Finin, University of Maryland, USA
Fausto Giunchiglia, University of Trento, Italy
Michael Gruninger, NIST, USA
Ramanathan Guha, IBM Research, USA
Frank van Harmelen, Vrije Universiteit Amsterdam, Netherlands
Yannis Kalfoglou, University of Southampton, UK
Vipul Kashyap, Clinical Informatics R&D, USA
Alain Leger, France Telecom R&D, France
Maurizio Marchese, University of Trento, Italy
Deborah L. McGuinness, Stanford University, USA
John Mylopoulos, University of Toronto, Canada
Natalya Noy, Stanford University, USA
Leo Obrst, MITRE, USA
Dimitris Plexousakis, University of Crete, Greece
Michel Plu, France Telecom R&D, France
Fano Ramparany, France Telecom R&D, France
Luciano Serafini, ITC-IRST, Italy
Amit Sheth, University of Georgia, USA
Pavel Shvaiko, University of Trento, Italy
Heiner Stuckenschmidt, Vrije Universiteit Amsterdam, Netherlands
York Sure, University of Karlsruhe, Germany
Michael Uschold, Boeing, USA
Holger Wache, Vrije Universiteit Amsterdam, Netherlands
Christopher Welty, IBM Research, USA

Contents

PART 2—Posters and Statements of Interest

Multi-dimensional Ontology Views via Contexts in the ECOIN Semantic Interoperability Framework

Aykut Firat
Northeastern University
Boston, MA 02115, USA
a.firat@neu.edu

Stuart Madnick
Massachusetts Institute of Technology
Cambridge, MA 02142, USA
smadnick@mit.edu

Frank Manola
Independent Consultant
Wilmington, MA 01887, USA
fmanola@acm.org

Abstract

This paper describes the coupling of contexts and ontologies for semantic integration in the ECOIN semantic interoperability framework. Ontological terms in ECOIN correspond to multiple related meanings in different contexts. Each ontology includes a context model that describes how a generic ontological term can be modified according to contextual choices to acquire specialized meanings. Although the basic ECOIN concepts have been presented in the past, this paper is the first to show how ECOIN addresses the case of "single-ontology with multiple contexts" with an example of semantic integration using our new prototype implementation.

Introduction

With the globalization of information over the internet, achieving semantic interoperability among heterogeneous and autonomous systems has become an increasingly important endeavor. A key issue in applying ontologies in practical semantic interoperability problems has proven to be reducing the amount of work needed to agree on a shared model, to describe the different assumptions made by sources and receivers, and to express (or generate) the mappings required to transform the data when moving it between different sources and receivers. In this paper, we discuss the ECOIN[1] approach and how it is able to address this important issue.

In the ECOIN semantic interoperability framework, ontologies describe both the shared domain model and the ways in which contexts can specialize the shared model. This is done by providing a terminology with *generic* meanings, which are *modified* in local contexts to express *specialized* meanings. A *context model* coupled with the shared model explicitly specifies possible modification dimensions of an ontological term. For example, the meaning (and representation) of a generic term like *airfare* can be modified along the *currency*, *coverage*, and *inclusion* dimensions. A context, then, expresses the specific specializations of the shared model that define a given local model (and hence a local model is described by the combination of the shared model and a particular context.) In the airfare example, for instance, the meaning of *airfare* objects is made explicit by local sources when they specify the currency used (e.g. USD); and declare whether the coverage is one-way or round-trip and what is included in the airfare (e.g. tax and shipping).

In the rest of this paper, we first review the ECOIN framework. We then elaborate on this approach in more detail by using a practical example from the air-travel domain and continue with a brief overview of the related work which is contrasted with the ECOIN approach. Finally, we discuss the benefits of our approach in reducing the amount of work needed to (a) construct a shared model, (b) describe local models, and (c) express mappings between contexts.

[1] ECOIN stands for **E**xtended **Co**ntext **In**terchange (see [Firat, 2003]).

The ECOIN Framework

The ECOIN *framework* is a generic logic-based data model that provides a template for the integration of heterogeneous data sources. This template is defined as follows:

Definition: (ECOIN Framework)
An ECOIN framework is a quadruple **(O, S, C, M)** where each component is a set of logical predicates. **O** corresponds to *ontology* that includes both the *domain and context model*; **S** corresponds to *source declarations*; **C** corresponds to *context (instances)*; and **M** corresponds to *mappings* (*conversion function network*) defined between contexts.

In this framework, sources **(S)** and contexts **(C)** are described with respect to the ontology **(O).** Mappings **(M)** are structured according to the context model to enable translation between different contexts. Below each component is described in detail.

Ontology

Ontology in ECOIN includes both the domain and context model. As in other data integration frameworks, an ECOIN domain model is used to define a common type system for the application domain (e.g., financial analysis, travel

information) corresponding to the data sources that are to be integrated. Like many other conceptual models, an ECOIN domain model consists of a collection of (object) *types,* which may be related in a subtype hierarchy. Types have *attributes* to represent both the individual properties of objects and relationships between objects (both things and their properties are uniformly represented as objects).

The types in an ECOIN domain model are *semantic types*, in that they represent the *generic* semantics of the concepts used in the various data sources. A semantic type is impartial to the exact representation or meaning of its instances in specific contexts and encapsulates all. The various specializations of these concepts used by different sources or receivers are described using a special kind of property called a *modifier*. The modifiers in an ontology are chosen to explicitly describe the contextual

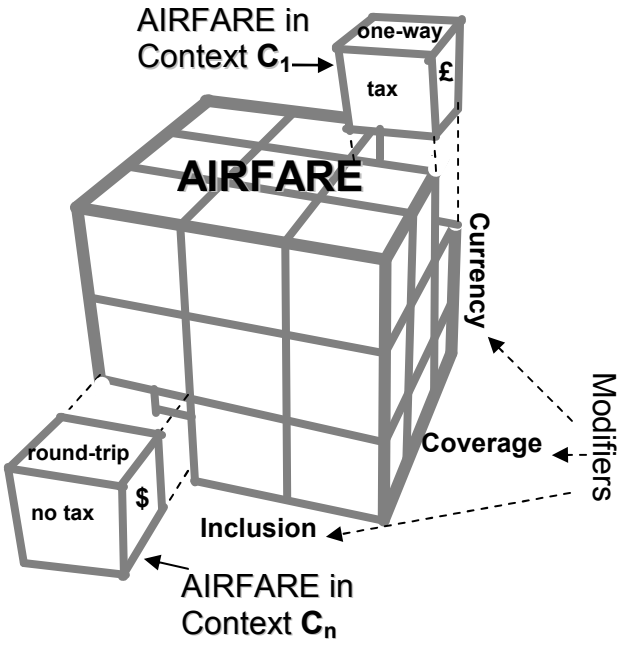


Figure 1 Multi-dimensional modification of the ontological term airfare

specializations of the generic types used by the sources and receivers. For example, in Figure 1 the generic ontological term *airfare* represented by the large cube can be specialized along three modification dimensions of *{Coverage, Currency, Inclusion}*. Different values of these modifiers identify the different component cubes of the overall airfare cube.

The *modifiers* in an ontology collectively define its *context model*; and the collection of *modifier objects* that describe the specializations that can be made by a given source or receiver defines its *context*. Context declarations are source independent, thus multiple sources or receivers may use the same context (use the same specializations for various values), but often different sources use different contexts.

Modifiers themselves are semantic types, thus can be subject to specialization (e.g. how do you represent currency? USD vs. $.) This can be handled via defining modifiers of modifiers. In Figure 2, this situation is illustrated by a *CurrencyFormat* modifier for the *Currency*

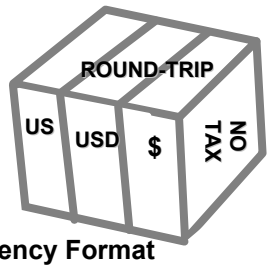


Figure 2 Modifiers themselves can be modified

modifier.

For objects without modifiers, the context model implies a *current* existence of a common representation and meaning across the sources and receivers. If this assumption changes at a later time, new modifiers can be introduced, further slicing and dicing the generic concepts.

In Figure 3, we illustrate a simplified ontology for the air travel domain. The domain and context model corresponding to the figure are represented in ECOIN with the following logical predicates: (The omitted predicates are indicated with three dots.)

Ontology

Domain Model

Types:
semanticType(country). semanticType(idType)….
semanticType(coverageType).

Type hierarchy:
isa(airfare, moneyAmt). isa(tax, moneyAmt).

Attributes/Relationships:
cxnCountry(ticket,country),….,hasID(ticket, idType).

Context Model

Modifiers:
lformat(airport, Context, airportFormat).
dformat(date, Context, dateFormat).
inclusion(airfare, Context, inclusionType).
coverage(airfare, Context, coverageType).
currency(moneyAmt, Context, currencyType).

The variable 'Context' in the Context Model signifies that a modifier is defined with respect to a given context, thus may acquire different values in different contexts.

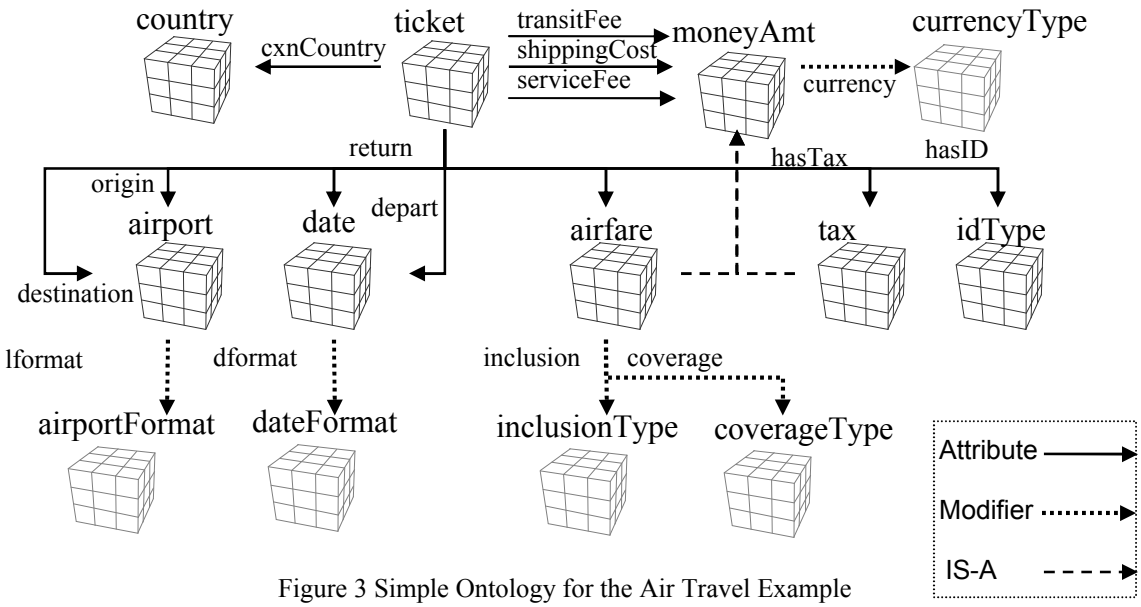


Figure 3 Simple Ontology for the Air Travel Example

Sources

Sources in the ECOIN framework are uniformly treated as relational sources (i.e., as having relational schemas). Many non-relational sources, such as HTML and XML web sites and web services, can be transformed into relational sources via wrappers [Firat at al. 2000]. A wrapped web source, for example, can be represented in logical predicates as (refer to Figure 5):

cheaptickets(Id, Airline, Price, Tax, DepDate, ArrDate, DepCity, CxnCountry, ArrCity)

In the ECOIN framework, these are called *primitive relations*, because these sources are not yet tied to an ontology. These primitive relations are elevated into *semantic relations* by annotating the semantic type and context of each primitive relation.
The semantic relation cheaptickets' can then be expressed as follows[2]:

cheaptickets'(Id', Airline', Price', Tax', Depdate', ArrDate', DepCity', CxnCountry',ArrCity') ←
Id'=object(idType, Id, c_ct, cheaptickets(Id, Airline, Price, Tax,Depdate,ArrDate, DepCity, CxnCountry, ArrCity)), …,ArrCity' = object(…).

With this elevation each column of the cheaptickets relation is indirectly tied to the air travel ontology. For the Id column, for instance, this is accomplished by associating the value Id with the idType semantic type in the cheaptickets context c_ct. Id' in the above declaration is the semantic object corresponding to the primitive object Id from the cheaptickets relation.

[2] Notation: We add a single quote ' to semantic objects/relations to distinguish them from primitive ones.

In addition, the attribute relationships defined by the ontology are instantiated as part of source declarations. For example, the cxnCountry relationship would be declared for this source as follows:

cxnCountry(T',C')←cheaptickets'(T',_,_,_,_,_,_,C',_).[3]

This declaration says that the cxnCountry of a semantic object T' is another semantic object C', both of which can be obtained from the semantic relation cheaptickets'. This is also known as the Global As View (GAV) approach of relating sources and the global model.

Context (Instances)

For *sources*, contexts define the specializations used for the underlying data values; and for *receivers* contexts describe the specializations assumed in viewing the data values. These specializations may be about the representation of data (e.g. European vs. American style date formats) or nuances in meaning (e.g. nominal vs. bottom-line prices).

To define a source or receiver context, modifier assignments need to be made. For example, the context labeled as c_ct can be described with the following predicates :

currency(Airfare', c_ct, Currency') ←
transitFee(Ticket',Airfare'),
cxnCountry(Ticket',Country') ,
countryCurrency(Country', Currency').[4]

currency(MoneyAmt', c_ct, Currency') ← Currency' = object(currencyType, "USD", c_ct, constant("USD")).

[3] Underscores, as in Prolog, are used to designate any value.
[4] Here, countryCurrency is a semantic relation that relates countries and currencies.

inclusion(Airfare', c_ct, Inclusion') ← Inclusion' = object(inclusionType, "nominal", c_ct, constant("nominal")).

coverage(Airfare', c_ct, Coverage') ← Coverage' = object (coverageType, "oneway", c_ ct, constant("oneway")).

lformat(Airport', c_ct, LFormat')← LFormat' = object(airportFormat, "airport", c_ct, constant("airport")).

dformat(Date', c_ct, DFormat') ← DFormat' = object (dateFormat, "American", c_ct, constant("American")).

These modifier declarations, which use attribute declarations, semantic relations, and some other constructs, explicitly specify which view of the ontology is adopted by the cheaptickets source. Accordingly, the ontology corresponding to the cheaptickets source treats *airfare* as *the one-way nominal price of a ticket in US dollars*. Currency in the cheaptickets context is US dollars except for transitFees which are given in the currency of the transit country. The arrival and departure locations are expressed as airport codes, and date is given using the *American style*.

Mappings

Mappings in ECOIN ensure that a view of the ontology adopted in a context is appropriately mapped to a corresponding ontological view in another context. This is accomplished by defining a conversion function network for each ontological term. Conversion functions are *atomically* defined for each modifier dimension as illustrated in Figure 4.

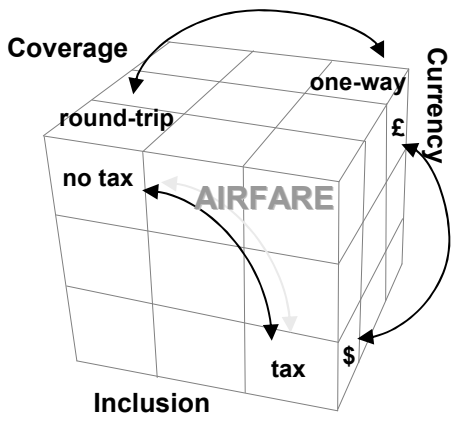


Figure 4 Organization of Conversion Functions for the Ontological term Airfare

As an example, the conversion function for the currency modifier dimension is encoded declaratively in terms of logical predicates as follows:

$f_{currency}$(X, VS, SC, VCurrencyS, VCurrencyT, TC, VT) ←
value(Today, SC, VToday), system_date(VToday),
value(CurrencyS,SC,VCurrencyS),
value(CurrencyT,SC,VCurrencyT),
currencyrates'(CurrencyS,CurrencyT, Today, Rate),
value(Rate, SC, VRate), mul(VS, VRate, VT).

For semantic airfare objects, this function uses the modifier value VCurrencyS in source context SC, and modifier value VCurrencyT in target context TC to translate the source value VS of semantic object X to value VT in target context. The value(A,C,B) predicate used above is read as "the value of semantic object A in context C is B". The function is also using another semantic relation currencyrates'; a system function system_date(VToday) and an arithmetic predicate mul to express multiplication.

As in the currency conversion function example above, conversion functions can sometimes be defined parametrically, thus may cover all of the modifier value pairs with a single function. When this can not be done, conversion functions can be defined as a network to minimize the number of declarations, leaving the tasks of combining, inverting, and simplifying to the mediator. Furthermore, most conversion functions are orthogonal, i.e. they can be applied in any order. When they are not orthogonal, priorities are used to determine the order they are to be executed. The details of conversion function network organization can be found in [Firat et al. 2005].

Practical Application

Consider the simplified scenario shown in Figure 5 having a single source *cheaptickets* and a single receiver (user) with conflicting assumptions. (This scenario, including the technical details of query mediation, is discussed more thoroughly in [Firat et al. 2005].) Surprisingly, even the semantic differences between a single source and a receiver provide enough complexity to highlight some of the interesting issues. Under this scenario, the user is an international student looking for a round trip airfare from Boston to Istanbul, with departure on June 1st and return on August 1st 2004. He wants to obtain the price and airline information for his trip and formulates the following SQL query Q1 using column names from the source:

Q1: SELECT Airline, Price
FROM CheapTickets
WHERE DepDate = "01/**06**/04" and ArrDate = "01/**08**/04" and DepCity = "**Boston**" and ArrCity = "**Istanbul**";

As a result of the contextual differences illustrated in Figure 5, without any mediation the user's query would return an empty answer, because cheaptickets has city codes instead of city names; and dates are in American format (refer to sample data). Even if these specific

Context of User

* Fares are expected to be bottom-line price
(round trip, includes taxes, ticket shipment, and transit fees)
* Date is expressed in European style (dd/mm/yy)
* Departure and Destination locations are expressed as city names
* Currency is US $
* Today's date: 01/05/04


```
SELECT Airline, Price
FROM cheaptickets
WHERE DepDate = "01/06/04" and
ArrDate= "01/08/04" and DepCity= "Boston"
and ArrCity= "Istanbul";
```

Context of cheaptickets

* All fares are for each way of travel and do not include fees and taxes.
* Date is expressed in American style (mm/dd/yy)
* Departure and Destination locations are expressed as three letter airport codes
* Currency is US dollars except for transit fees, which are in the currency of the country that issues the fee.
* Direct air transit fee of £27 is applied if the plane has a connecting flight from United Kingdom
* Ticket shipping cost is $20
* Service fee of $5 is charged

cheaptickets

ID (I)	Airline (A)	Price (P)	Tax (T)	DepDate (DD)	ArrDate (AD)	DepCity (DC)	CxnCountry (CC)	ArrCity (AC)
1	British Airways	495	75	06/01/04	08/01/04	BOS	United Kingdom	IST
2	Lufthansa	525	79	06/01/04	08/01/04	BOS	Germany	IST
…	…	…	…	…	…	…	…	…

Context of Ancillary Sources

Date is expressed in American style

currencyrates

FromCur	ToCur	ExchangeRate	Date
£	$	1.75	05/10/04
…	…	…	…

cityairport

City	Airport
Boston	BOS
Istanbul	IST
…	…

Figure 5 Airfare Example Scenario

differences were dealt with, for example by writing a new query Q2 with changed city codes and date formats (which itself might be a significant challenge for the user, especially if unfamiliar with the details of each of the multiple sources involved):

```
Q2: SELECT Airline, Price
FROM CheapTickets
WHERE DepDate = "06/01/04" and ArrDate = "08/01/04"
and DepCity = "BOS" and ArrCity = "IST";
```

the results returned would be:

Airline	Price
British Airways	495
Lufthansa	525

which is not the correct bottom line price the user expects. Given these results, the user may incorrectly think that British Airways is the cheaper option. If the original query Q1 were submitted to the ECOIN system, however, the semantic conflicts between the sources and the receiver would be automatically determined and reconciled, and Q1 would be rewritten into the following mediated query:

```
MQ1:SELECT Airline, 2*(Price+Tax+27*exchangeRate)+25
FROM cheaptickets, currencyrates,
(select Airport from cityairport where city="Boston") dCode,
(select Airport from cityairport where city="Istanbul") aCode
WHERE DepDate = "06/01/04" and ArrDate="08/01/04"
and DepCity= dCode.Airport and ArrCity=aCode.Airport
and CxnCountry= "United Kingdom" and fromCur= "GBR"
and toCur= "USD" and Date= "05/10/04";
UNION
SELECT Airline, 2 * (Price+Tax) +25
FROM cheaptickets,
(select Airport from cityAirport where city="Boston") dCode,
(select Airport from cityAirport where city="Istanbul") aCode
WHERE DepDate = "06/01/04" and ArrDate="08/01/04"
and DepCity= dCode.Airport and ArrCity=aCode.Airport
and CxnCountry <> "United Kingdom";
```

In the mediated query MQ1, in addition to representational conflicts such as format differences in date and city codes, semantic conflicts in the interpretation of airfare (price) are also resolved. Mediating such semantic conflicts involves creating a conflict table by comparing the modifiers involved in the query; identifying which mappings to use from the conversion function network to resolve the

conflicts; and applying symbolic equation solving techniques to a number of equational relations for inversion, composition and simplification under the Abductive Constraint Logic Programming framework [Firat et. al. 2005].

The ECOIN system further processes this mediated query by an optimizer to produce an efficient plan, and executes it by a query processor, which submits subqueries to individual sources that can optimally execute the subqueries and perform the data transformations. The final results reported by the system below now allow the user to make the right choice and choose Lufthansa over British Airways:

Airline	Price
British Airways	1260
Lufthansa	1233

As this practical application shows, despite sharing the same ontology, the users and sources are not locked into a single integrated view. Multiple integrated views can co-exist with a well defined context model coupled with the ontology.

Related Work and Discussion

One of the fundamental issues of information integration is achieving interoperability between multiple local models. There have been various approaches proposed in the past. Although there are some similarities, ECOIN has a number of important distinct differences and advantages.

In database integration, for instance, local models are in the form of database schemas and achieving interoperability among multiple schemas constitutes the fundamental problem. The traditional centralized solution maps local models (schemas) to a shared standard ontology (global schema) to eliminate representational and semantic disparities. This approach has been criticized for lack of scalability and difficulty of maintenance over time. Furthermore, it is seen as overly restrictive and inflexible in trying to reconcile local models that suit different needs in a single shared model [Bouquet and Serafini 04].

Modularized versions of the traditional centralized approach with the explicit use of "contexts" appears in [McCarthy and Buvac 97], and in CYC [Lenat et al., 1990; Guha, 1991]. In these approaches, axioms and statements are true only in a context. This is expressed by a *modality*[5] called ist(c,p)[6]. For example,

c_0: ist(context-of("Sherlock Holmes stories"), "Holmes is a detective").

means that the statement "Holmes is a detective" is true in the context of Sherlock Holmes stories. The preceding c_0 denotes that this statement is asserted in an outer context, thus points out to the nested composition of context dependent statements. Formulas between contexts can be related together with the use of lifting axioms. In [McCarthy and Buvac 97], an example of integrating Navy and General Electric (GE) databases, which differ on the definition of engine prices, is given. In the Navy database price includes assortment of spare parts and warranty, whereas in GE price is the plain engine price. Contexts defined in this example are c_{GE}, c_{navy} corresponding to the GE and navy databases and c_{ps}, the problem solving context. The details of this example are shown in Table 1 (i.e. the query posed in the problem solving context, the existing facts expressed in their own context, and lifting axioms that define translations between different contexts).

Query

c_{ps}: ist(c_{navy}, price(FX-22-engine, $3611K))

Facts

ist(c_{GE}, price(FX-22-engine, $3600K)).
ist(c_{GE}, price(FX-22-engine-fan-blades, $5K)).
ist(c_{GE}, price(FX-22-engine-two-year-warranty, $6K)).
ist(c_{navy},spares(FX-22-engine,FX-22-engine-fan-blades)).
ist(c_{navy}, warranty(FX-22-engine,FX-22-engine-two-year-warranty)).

Lifting axioms

value[7](c_{GE}, price(x)) = GE-price(x)
value(c_{navy}, price(x)) = GE-price(x) + GE-price(spares($C_{Navy,}$ x)) + GE-price(warranty(C_{Navy}, x)).

Table 1 Navy and General Electric Integration Example

An opposite approach, called "compose and conquer" [Bouquet et. al. 01], is based on the premise that the existence of a global ontology is not viable in open settings such as the envisioned Semantic Web. In the proposed solution, relations between local models are established on a peer-to-peer basis, as a collection of constraints on what can (or cannot) be true in a local model given that there is some relation with what holds in another local model. This approach has been used in [Ghidini and Serafini 98, 00], in integrating information systems. They provide an example that integrates the databases of four fruit sellers with different contexts. Conflict resolution between contexts is done pair wise for each database, since they do not subscribe to a common global theory. In the example, one of the sellers (1) provides fruit prices without including taxes, the other denoted as the mediator (m) considers prices with taxes (7% percent). This conflict is resolved by defining a view constraint as following:

1: has-price(x,y) → m:∃y′ has-price(x,y′) ∧ y′ =y +(0.07*y)

[5] The classification of propositions on the basis of whether they assert or deny the possibility, impossibility, contingency, or necessity of their content.

[6] Read as "p is true in context c"

[7] value(c,t) is a function which returns the value of term t in context c

This view constraint establishes the link between differing price definitions of source 1 and mediator m.

While this approach offers important benefits, especially in providing a viable architecture for open settings, the lack of a shared model creates a number of serious problems. Even finding a way to query disparate data sources, connected on a peer-to-peer basis, becomes a non-trivial task. Furthermore, the coordination of establishing relationships between local models on a peer-to-peer basis is problematic. ECOIN provides a much simpler solution.

Contextual Coupling of Ontology and Local Models in ECOIN

The ECOIN strategy of relating local models favors the use of ontologies to relate local models, albeit in a much more flexible way than the traditional centralized approaches. It may be too early to predict how the Semantic Web will ultimately evolve, but it is perceivable that similar local models will be linked via ontologies, which in turn may be treated as local models and linked via higher level ontologies thus achieving gradual semantic interoperability. Given such a possibility, the ECOIN approach introduces a contextual coupling of ontology and local models.

Our approach may seem similar to the efforts discussed in [McCarthy and Buvac 97], [Lenat et al., 1990; Guha, 1991], [Kashyap and Sheth 96], [Bouquet et al. 2004] at the surface level, but there are important differences. While we like the explicit treatment of contexts in these efforts; and share their concern for sustaining an infrastructure for data integration, our realization of these differ significantly. First, the ontology in ECOIN only defines *generic* terms without specifying their exact semantics, which has no equivalent in the aforementioned approaches. Second, *lifting axioms* [Guha 1991] in our case operate at a finer level of granularity: rather than writing axioms which map "statements" present in a data source to a common knowledge base, they are used for translating "properties" of individual "data objects" and organized as a conversion function network between contexts. These differences account largely for the scalability and extensibility of our approach.

Compared with the Context-OWL (C-OWL) approach discussed in [Bouquet et al. 2004], our effort is more focused on query mediation than trying to come up with a general theory of contextual reasoning. Furthermore, the contextual mappings in our case go beyond the rather limited set of mappings that exist in C-OWL (i.e. equivalent, onto (superset), into (subset), compatible, and incompatible). The limited expressiveness of the C-OWL language fails to address the contextual differences found in most practical settings.

The *description logic* based context representation as contextual coordinates in [Kashyap and Sheth 96] has compelling similarities with our approach. While the desire to dynamically express the context of data is paramount in both approaches, there are also important conceptualization differences. While the *contextual coordinates* denote aspects of the *context* in [Kashyap and Sheth 96], *modifiers* denote aspects of an *ontological term* in ECOIN. Our conceptualization results in a simpler context model, which works very well for query mediation by allowing us to organize, compose, invert and simplify conversion functions that maps between different contexts.

Compared with the "compose and conquer" approach, the ECOIN approach is similar in its desire to perform peer to peer mappings (although mappings need not be defined between every peer). Unlike "compose and conquer", however, ECOIN assumes the existence of an ontology to tie the sources, but this ontology does not act like a "global schema". The ontology acknowledges the *minimal* agreements between the local models, and defines a well-defined (yet extensible) context model to facilitate the reconciliation of possible conflicts between local models.

With these differences in mind, the benefits of the ECOIN approach can be summarized as follows.

First, ontology developers do not have to standardize the exact meaning and representation of ontological terms; but only need to agree on generic identities without exposing the specific details. A major advantage of this approach is that ontology developers frequently find it straightforward (if not necessarily "easy") to agree on the generic concepts; it is getting all the precise details worked out that creates a lot of the work. Moreover, it's often the case that differences in these precise details are only discovered later (sometimes even after the system is in operation). The ECOIN approach enables these details to be factored out, reducing the amount of work needed to introduce these details all at once.

Second, allowing the same ontological term to assume nuances of meaning and varying representations in local contexts saves the ontology from being cluttered with non-essential terms such as *airfareIn$*, *airfareWithTax*, *airfareRoundTrip*, etc. In ECOIN, only the essential term *airfare* with its modification dimensions belongs to the ontology, and takes its specialized meanings in local contexts with corresponding modifier values.

Third, because the ontology is impartial to the precise semantics defined in the various contexts, mappings are not defined between the sources and the ontology as it is done in most current approaches to information integration. Instead, mappings are structured with respect to a context model and defined for each modification dimension as a *conversion function network*. This modularization of mappings allows a mediator to create custom point to point translations between contexts by selecting or composing appropriate mappings from the conversion function network. These capabilities of ECOIN have been demonstrated in an example application requiring the integration of counter-terrorism intelligence information, where ECOIN was able to generate over 22,000 conversion programs to enable semantic integration amongst 150 data sources and receivers using just six parameterized conversion rules [Zhu and Madnick 2004].

Conclusion

The ECOIN semantic information integration framework couples ontologies and contexts in a unique way for the semantic integration of disparate data sources. The approach presupposes the existence of an ontology, but unlike traditional approaches this ontology does not provide a rigid specification of the meanings and representations of its terms. In this novel conceptualization, ontological terms are modified according to a context model, thus correspond to multiple integrated views according to contextual choices.

We have implemented these ideas in a prototype [Firat 2003] using the Eclipse Prolog engine [Cheadle et al. 2003] and procedural programming languages. This prototype provides mediated access to traditional databases, as well as semi-structured web sites, and web services, creates and maintains metadata that are used in ECOIN through graphical interfaces, and supports merging multiple applications.

We believe that semantic information integration should have the dual purpose of: (1) *reconciling semantic heterogeneity across information sources*; and (2) *supporting semantic heterogeneity across information receivers*. The ECOIN approach achieves this objective by providing an interoperability framework that requires minimal agreement on a generic ontology, and allowing the local models to modify the ontology to fit their context.

References

Bouquet P., and Serafini, L. 2004. Meaning Coordination and Negotiation. *In Working Notes of the ISWC Workshop on Meaning Coordination and Negotiation*, *3rd International Semantic Web Conference*, Hiroshima, Japan.

Bouquet P., Ghidini C., Giunchiglia F., and Blanzieri E. 2001. Theories And Uses Of Context In Knowledge Representation And Reasoning. Technical Report # 0110-28, Istituto Trentino di Cultura.

Bouquet P., and Serafini L. 2003. On the Difference between Bridge Rules and Lifting Axioms. *In Proceedings of Modeling and Using Context, 4th International and Interdisciplinary Conference*, CONTEXT: 80-93

Bouquet P., Giunchiglia F., Harmelen, F., Serafini, L., and Stuckenschmidt, H. 2004. *Contextualizing Ontologies*, Journal of Web Semantics , vol. 26, 2004: 1-19.

Cheadle, A. M., Harvey, W., Sadler, A.J., Schimpf, J., Shen, K., and Wallace M. G. 2003. ECLiPSe: An Introduction, Imperial College London, Technical Report ICParc-03-1.

Firat, A. Madnick S., Siegel M., Grosof, B., and Manola, F. 2005 *Reconciling Semantic Heterogeneity with Symbolic Equation Solving Techniques*. Forthcoming.

Firat, A. 2003. *Information Integration using Contextual Knowledge and Ontology Merging*, Ph.D. Thesis, Massachusetts Institute of Technology.

Firat, A., Madnick, S., and Siegel, M. 2000. The Caméléon Web Wrapper Engine, *In Proceedings of the VLDB2000 Workshop on Technologies for E-Services*, 1-9.

Ghidini, C., and Giunchiglia, F. 2001. *Local Models Semantics, or Contextual Reasoning = Locality + Compatibility*. Artificial Intelligence. 127(2):221-259.

Ghidini, C., and Serafini, L. 1998. Information Integration for Electronic Commerce. *In Agent Mediated Electronic Commerce. First International Workshop on Agent Mediated Electronic Trading*, AMET-98, Volume 1571 of LNAI. Springer.

Guha R. V. (1991). Contexts: a formalization and some applications, MCC Tech Rep ACT-CYC42391.

Kashyap, V.; Sheth, A.P. 1996. *Semantic and Schematic Similarities between Database Objects: A Context-Based Approach*, VLDB Journal 5(4):276-304.

Lenat, D., R. V. Guha, K. Pittman, D. Pratt, and M. Shepherd. 1990. *Cyc: Towards programs with common sense*. Communications of the ACM 33(8).

McCarthy, John and Buvac, S, 1997. *Formalizing context (expanded notes)*. In: Aliseda, A., van Glabbeek, R. and Westerstrahl, D., Editors, 1997. Computing natural language, Center for the Study of Language and Information, Stanford, CA.

Zhu, H., and Madnick, S. 2004. Context Interchange as a Scalable Solution to Interoperating Amongst Heterogeneous Dynamic Services. *In Proceedings of the Third Workshop on eBusiness* (Web2004) .

Putting Things in Context: A Topological Approach to Mapping Contexts and Ontologies

Aviv Segev, Avigdor Gal
Technion - Israel Institute of Technology
Haifa 32000
Israel
{asegev@tx, avigal@ie}.technion.ac.il

Abstract

Ontologies and contexts are complementary disciplines for modeling views. In the area of information integration, ontologies may be viewed as the outcome of a manual effort of modeling a domain, while contexts are system generated models. In this work, we aim at formalizing the inter-relationships between a manually generated ontology and automatically generated contexts. We provide a formal mathematical framework that delineates the relationship between contexts and ontologies. We then use the model to define the uncertainty associated with automatic context extraction from existing documents and provide a ranking method, which ranks ontology concepts according to their suitability with a given context. Throughout this work we motivate our research using QUALEG, a European IST project that aims at providing local government an effective tool for bi-directional communication with citizens.

Keywords: Ontology, Context, Topology mapping

Introduction

Ontologies and contexts are both used to model views, which are different perspectives of a domain. Some consider ontologies as shared models of a domain and contexts as local views of a domain. In the area of information integration, an orthogonal classification exists, in which ontologies are considered a result of a manual effort of modeling a domain, while contexts are system generated models (Segev, Leshno, & Zviran 2004). As an example, consider an organizational scenario in which an organization (such as a local government) is modeled with a global ontology. A task of document classification, in which new documents are classified upon arrival to relevant departments, can be modeled as an integration of contexts (automatically generated from documents) into an existing ontology. A simple example of a context in this setting would be a set of words, extracted from the document.

This approach was recently taken in QUALEG, a European Commission project aimed at increasing citizen participation in the democratic process.[1] In QUALEG, contexts are used to specify the input from citizens and then to provide services - routing emails to departments, opinion analysis on topics at the forefront of public debates, and the identification of new topics on the public agenda.

The two classifications are not necessarily at odds. In the example given above, documents may be email messages from citizens, expressing a local view of a domain. Yet, the classification of manual vs. automatic modeling of a domain has been the center of attention in the area of data integration and schema matching in the past few years. In particular, many heuristics were proposed for the automatic matching of schemata (*e.g.,* Cupid (Madhavan, Bernstein, & Rahm 2001), GLUE (Doan *et al.* 2002), and OntoBuilder (Gal *et al.* 2005b)), and several theoretical models were proposed to represent various aspects of the matching process (Madhavan *et al.* 2002; Melnik 2004; Gal *et al.* 2005a).

In this work, we aim at formalizing the inter-relationships between an ontology, a manually generated domain model, and contexts, partial and automatically generated local views. We provide a formal mathematical framework that delineates the relationships between contexts and ontologies. Following the motivation given above, we discuss the uncertainty associated with automatic context extraction from existing documents and provide a ranking model, which ranks ontology concepts according to their suitability with a given context. We provide examples from the QUALEG project.

Our contributions are as follows:

- We present a framework for combining contexts and ontologies using topological structures, and model the uncertainty inherent to automatic context extraction.
- We provide a model for ranking ontology concepts relative to a context.
- Using real world scenario, taken from email messages from citizens in a local government, we demonstrate three tasks that involve mapping contexts to ontologies, namely email routing, opinion analysis, and public agenda identification.

The rest of the paper is organized as follows. We first discuss related work on the topic. Next, we propose a model for combining contexts and ontologies and present a ranking model to map contexts to ontologies. The last section includes concluding remarks and suggestions for future work.

[1] http://www.qualeg.eupm.net/

Related Work

This section describes related work in four different research areas, namely context representation, ontologies, context extraction, and topologies.

Context Representation

The context model we use is based on the definition of context as first class objects formulated by McCarthy (McCarthy 1993). McCarthy defines a relation $ist(\mathcal{C}, P)$, asserting that a proposition P is true in a context $\mathcal{C}$. We shall use this relation when discussing context extraction.

It has been proposed to use a multilevel semantic network to represent knowledge within several levels of contexts (Terziyan & Puuronen 2000). The zero level of representation is a semantic network that includes knowledge about basic domain objects and their relations. The first level of representation uses a semantic network to represent contexts and their relationships. The second level presents relationships of metacontexts, the next level describes metametacontext, and so on and so forth. The top level includes knowledge that is considered to be true in all contexts. In this work we do not explicitly limit the number of levels in the sematic network. However, due to the limited capabilities of context extraction tools nowadays (see below), we define context as sets of sets of descriptors at zero level only and the mapping between contexts and ontology concepts is represented at level 1. Generally speaking, our model requires $n + 1$ levels of abstraction, where n represents the abstraction levels needed to represent contexts and their relationships.

Previous work on contexts (Siegel & Madnick 1991) uses metadata for semantic reconciliation. The database metadata dictionary (DMD) defines the semantic and assignment domains for each attribute and the set of rules that define the semantic assignments for each of these attributes. The application semantic view (ASV) contains the applications definition of the semantic and assignment domain and the set of rules defining the applications data semantic requirements. They define the semantic domain of an attribute T as the set of attributes used to define the semantics of T. Work by (Kashyap & Sheth 1996) use contexts that are organized as a meet semi-lattice and associated operations like the greatest lower bound for semantic similarity are defined. The context of comparison and the type of abstractions used to relate the two objects form the basis of a semantic taxonomy. They define ontology as the specification of a representational vocabulary for a shared domain of discourse. Both these approaches use ontological concepts for creating contextual descriptions and serve best when creating new ontologies. The approach proposed herein assumes the existence of an ontology to which contexts should be mapped. Another difference is that in (Kashyap & Sheth 1996), an ontology concept is taken to be the intersection of context sets, while we view ontology concepts as the union of context sets.

Ontology

Ontologies were defined and used in various research areas, including philosophy (where it was coined), artificial intelligence, information sciences, knowledge representation, object modeling, and most recently, eCommerce applications. In his seminal work, Bunge defines Ontology as a world of systems and provides a basic formalism for ontologies (Bunge 1979). Typically, ontologies are represented using a Description Logic (Borgida & Brachman 1993; Donini *et al.* 1996), where subsumption typifies the semantic relationship between terms; or Frame Logic (Kifer, Lausen, & Wu 1995), where a deductive inference system provides access to semi-structured data.

The realm of information science has produced an extensive body of literature and practice in ontology construction, *e.g.*, (Vickery 1966). Other undertakings, such as the DOGMA project (Spyns, Meersman, & Jarrar 2002), provide an engineering approach to ontology management. Work has been done in ontology learning, such as Text-To-Onto (Maedche & Staab 2001), Thematic Mapping (Chung *et al.* 2002), OntoMiner (H. Davulcu & Nagarajan 2003), and TexaMiner (Kashyap *et al.* 2005) to name a few. Finally, researchers in the field of knowledge representation have studied ontology interoperability, resulting in systems such as Chimaera (McGuinness *et al.* 2000) and Protègè (Noy & Musen 2000).

Our model is based on Bunge's terminology. We aim at formalizing the mapping between contexts and ontologies and provide an uncertainty management tool in the form of concept ranking. Therefore, in our model we assume an ontology is given, designed using any of the tools mentioned above.

Context Extraction

The creation of taxonomies from metadata (in XML/RDF) containing descriptions of learning resources was undertaken in (Papatheodorou, Vassiliou, & Simon 2002). Following the application of basic text normalization techniques, an index was built, observed as a graph with learning resources as nodes connected by arcs labeled by the index words common to their metadata files. A cluster mining algorithm is applied to this graph and then the controlled vocabulary is selected statistically. However, a manual effort is necessary to organize the resulting clusters into hierarchies. When dealing with medium-sized corpora (a few hundred thousand words), the terminological network is too vast for manual analysis, and it is necessary to use data analysis tools for processing. Therefore, Assadi (Assadi 1998) has employed a clustering tool that utilizes specialized data analysis functions and has clustered the terms in a terminological network to reduce its complexity. These clusters are then manually processed by a domain expert to either edit them or reject them.

Several distance metrics were proposed in the literature and can be applied to measure the quality of context extraction. Prior work had presented methods based on information retrieval techniques (van Rijsbergen 1979) for extracting contextual descriptions from data and evaluating the quality of the process. Motro and Rakov (Motro & Rakov 1998) proposed a standard for specifying the quality of databases based on the concepts of soundness and completeness. The method allowed the quality of answers to arbitrary

queries to be calculated from overall quality specifications of the database. Another approach (Mena *et al.* 2000) is based on estimating loss of information based on navigation of ontological terms. The measures for loss of information were based on metrics such as precision and recall on extensional information. These measures are used to select results having the desired quality of information.

To demonstrate our method, we propose later in this paper the use of a fully automatic context recognition algorithm that uses the Internet as a knowledge base and as a basis for clustering (Segev, Leshno, & Zviran 2004). Both the contexts and the ontology concepts are defined as topological sets, for which set distance presents iteself as a natural choice for a distance measure.

Topology

In recent years different researchers have applied principles from the mathematical domain of topology in different fields of Artificial Intelligence. One work uses topological localization and mapping for agent problem solving (Choset & Nagatani 2001). Other researchers have implemented topology in metrical information associated with actions (Shatkay & Kaelbling 1997; Koenig & Simmons 1996). In another method of topological mapping, which describes large scale static environments using a hybrid topological metric model, a global map is formed from a set of local maps organized in a topological structure, where each local map contains quantitative environment information using a local reference frame (Simhon & Dudek 1998). Remolina and Kuipers present a general theory of topological maps whereby sensory input, topological and local metrical information are combined to define the topological maps explaining such information (Remolina & Kuipers 2004).

In this work we use topologies as a tool of choice for integrating contexts and ontologies.

A Model of Context and Ontology

In this section we formally define contexts and ontologies and propose a topology-based model to specify the relationships between them.

Contexts and Ontologies

A *context* $\mathcal{C} = \left\{ \{\langle c_{ij}, w_{ij} \rangle\}_j \right\}_i$ is a set of finite set of descriptors c_{ij} from a domain $\mathcal{D}$ with appropriate weights w_{ij}, defining the importance of c_{ij}. For example, a context $\mathcal{C}$ may be a set of words (hence, $\mathcal{D}$ is a set of all possible character combinations) defining a document Doc, and the weights could represent the relevance of a descriptor to Doc. In classical Information Retrieval, $\langle c_{ij}, w_{ij} \rangle$ may represent the fact that the word c_{ij} is repeated w_{ij} times in Doc.

An *ontology* $O = (V, E)$ is a directed graph, with nodes representing concepts (*things* in Bunge's terminology (Bunge 1977; 1979)) and edges representing relationships. A single concept is represented by a name and a context $\mathcal{C}$. Figure 1 (top) displays the graphical representation of an ontology.

Example 1. *To illustrate contexts and ontologies, consider the local government of Saarbrücken. Two ontology concepts in the ontology of Saarbrücken are:*
(*Perspectives du Theatre,* $\{\{\langle$*Öffentlichkeitsarbeit, 2*$\rangle\}$, $\{\langle$*Multimedia, 1*$\rangle\}$, $\{\langle$*Kulturpolitik, 1*$\rangle\}$, $\{\langle$*Musik, 6*$\rangle\}$, ...$\}$)
and
(*Long Day School,* $\{\{\langle$*Förderbedarf, 1*$\rangle\}$, $\{\langle$*Mathematik, 2*$\rangle\}$, $\{\langle$*Musik, 2*$\rangle\}$, $\{\langle$*Interkulturell, 1*$\rangle\}\}$)
A context, which was generated from an email message using the algorithm in (Segev, Leshno, & Zviran 2004) (to be described later) is $\{\{\langle$*Musik,* $8\rangle\}$, $\{\langle$*Open Air,* $1\rangle\}\}$.

Modeling Context-Ontology Relationships

The relationships between ontologies and contexts can be modeled using topologies as follows. A *topological structure (topology)* in a set X is a collective family $\vartheta = (G_i / i \in I)$ of subsets of X satisfying
1. $J \subset I \Rightarrow \bigcup_{i \in J} G_i \in \vartheta$
2. J finite; $J \subset I \Rightarrow \bigcap_{i \in J} G_i \in \vartheta$
3. $\emptyset \in \vartheta, X \in \vartheta$

The pair (X,ϑ) is called a *topological space* and the sets in ϑ are called *open sets*. We now define a context to be an open set in a topology, representing a family ϑ of all possible contexts in some set X. Using the concrete example given above, let X be a set of sets of tuples $\langle c, w \rangle$, where c is a word (or words) in a dictionary and w is a weight. Note that ϑ is infinite since descriptors are not limited in their length and weights are taken from some infinite number set (such as the natural numbers $\mathbb{N}$).

A family $\mathbf{B} = (B_i / i \in I)$ is called a *filter base* (also known as a directed set, indexed set, or a base) if
1. $(\forall i) : B_i \neq \emptyset$
2. $(\forall i)\ (\forall j)\ (\exists k) : B_k \subset B_i \cap B_j$

A *filtered family* is a family of sets $(x_i / i \in I)$ associated with a filter base $\mathbf{B}$ on index I. A filtered family $(x_i) = (x(i) / i \in I, i \in \mathbf{B})$ forms a sequence of sets with the filter base.

We define a specific filtered family based on the concept of a context, as defined above. The definition is illustrated in Figure 2. Let Context Set A_1 define all the context sets that can be created out of one given context - this is only one context. Let Context Set A_2 be the sets of contexts that can be created from two given contexts. Context Set A_2 contains each of the contexts and the union of both contexts. This filtered family can continue expanding indefinitely.

Whenever a filtered family contains contexts that describe a single topic in the real world, such as school or festival, we would like to ensure that this set of contexts converges to one ontology concept v, representing this topic, *i.e.*, $A_n \rightarrow_{n \rightarrow \infty} v$. In topology theory, such a convergence is termed a *point of accumulation*, defined as follows.

Let A be a subset of a topological space X; An element $x \in X$ is a *point of accumulation* of the set A if every neighborhood $V(x)$ meets the set $A - x$, that is, if $x \in \overline{A - x}$. Figure 1(bottom) and Figure 2 illustrates ontology concepts as points of accumulation.

To illustrate the creation of an ontology concept let a context be a set containing a single descriptor

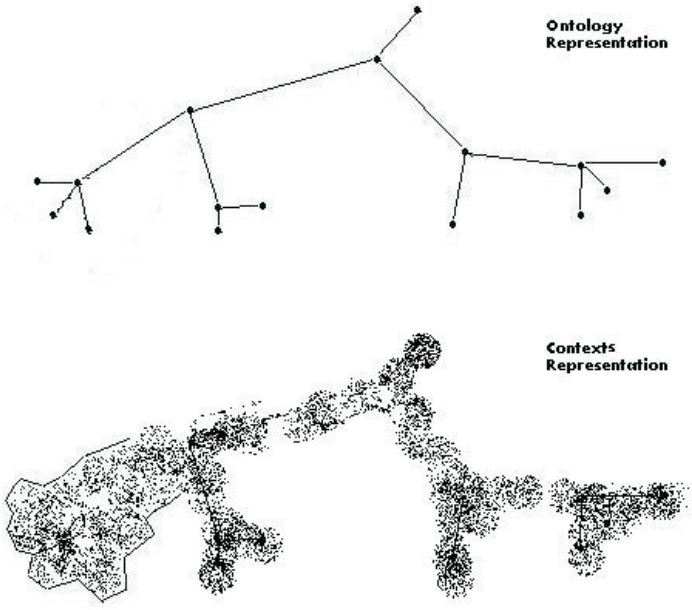


Figure 1: Contexts and Ontology Concepts

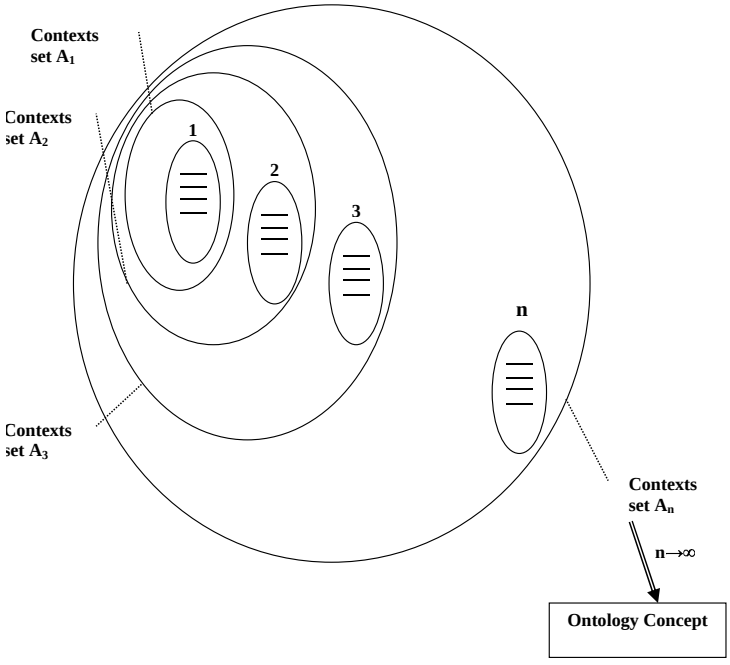


Figure 2: Contexts Sets Converging to Ontology Concept

$\{\langle$Mathematik, 2$\rangle\}$. If we add another context containing a single descriptor of $\{\langle$Musik, 2$\rangle\}$ we form a set of three contexts: $\{\{\langle$Mathematik, 2$\rangle\}$, $\{\langle$Musik, 2$\rangle\}$, $\{\langle$Mathematik, 2$\rangle$, $\langle$Musik, 2$\rangle\}\}$. As the possible sets of descriptors describing documents create an accumulating coverage, we can converge to an ontology concept, such as Long Day School, defined by a set, to which all the contexts set of descriptors belongs.

With infinite possible contexts, can we ensure the existence of a finite number of ontology concepts to which the contexts are mapped? As it turns out, such a guarantee exists in compact topologies. A topological space X is said to be *compact* if every family of open sets $(G_i/i \in I)$ forming a cover of X contains a finite subcover $\{G_{i_1}, G_{i_2}, ..., G_{i_n}\}$. That is, any collection of open sets whose union is the whole space has a finite subcollection whose union is still the whole space. The Bolzano-Weierstrass theorem (Berge 1997) ensures that if X is a compact space, every infinite subset A of X possesses a point of accumulation. Therefore, if the contexts' domain can be covered by a finite cover, such as the number of topics, we can be certain that any infinite set of contexts will accumulate to an ontology concept.

Discussion and Examples

A context can belong to multiple context sets, which in turn can converge to different ontology concepts. Thus, one context can belong to several ontology concepts simultaneously. For example, a context $\langle$Musik, 2$\rangle$ can be shared by many ontology concepts who has interest in culture (such as schools, after school institutes, non-profit organizations, *etc*.) yet it is not in their main role definition. Such overlap of contexts in ontology concepts affects the task of email routing. The appropriate interpretation of a context of an email that is part of several ontology concepts, is that the email is relevant to all such concepts. Therefore, it should be delivered to multiple departments in the local government.

Of particular interest are ontology concepts that are considered "close" under some distance metric. As an example, consider the task of opinion analysis. With opinion analysis, a system should judge not only the relevant area of interest of a given email, but also determine the opinion that is expressed in it. Consider an opinion analysis task, in which opinions are partitioned into two categories (*e.g.*, "for" and "against"). We can model such opinions using a common concept ontology (say, that of Perspectives du Theatre), with the addition of words that describe positive and negative opinions. An email whose context fit with the theme of Perspective du Theatre will be further analyzed to be correctly classified to the "for" or "against" bin. Opinion analysis can be extended to any number of opinions in the same way.

Earlier we have discussed the issue of topological space compactness and its impact on ontology generation. Since there are infinite number of contexts, it may be impossible to suggest a single ontology to which all concepts can be mapped. For local governments, shifting public agenda suggests that a notion of fixed ontology is not at all natural. Nevertheless, we would like to use the Bolzano-Weierstrass theorem to our benefit, and ensure that the contexts domain can be covered by a finite cover, to ensure the existence of points of accumulation.

From the discussion above, it is clear that a fixed ontology cannot serve as a solution. However, when taking a snapshot of a local government, ontology is fixed. Some aspects of the world are beyond the scope of the local government and if we add to the local government ontology a concept that represents all these aspects, we are ensured to have a finite cover of size $n + 1$, with n representing the concepts of current interest. Over time, emails that are beyond the current scope of the local government are accumulated under the $n + 1$ concept, and may be clustered to achieve a new point of accumulation, and thus a new topic of interest in the public agenda.

To summarize, the proposed model employs topological definitions to delineate the relationships between contexts and ontologies. A context is a set of descriptors and their corresponding weights. A filter base is a set of contexts that includes all of their possible unions. If the filter base has a point of accumulation to which the set of contexts converges, then it is defined as an ontology concept. The use of points of accumulations defines ontology concepts to be the union of contexts rather than intersection, as suggested in earlier works. We next turn our attention to the uncertainty inherent in automatic extraction of contexts.

Ranking Ontology Concepts

Up until now, the model we have provided assumed perfect knowledge in the sense that a context is a true representative of a local view and an ontology concept (and its related context) is a true representative of a global view. In the real world, however, this may not be the case. When a context is extracted automatically from some information source (*e.g.*, an email message), it may not be extracted accurately and descriptors may be erroneously added or eliminated. Also, even for manually crafted ontology concepts, a designer may err and provide an inaccurate context for a given concept.

In this section we highlight the uncertainty involved in automatic knowledge extraction and propose a method for managing such uncertainty. In particular, we discuss the impact of uncertainty on the three tasks presented above, namely email routing, opinion analysis, and public agenda.

Context Recognition Algorithms

Several methods were proposed in the literature for extracting context from text. A set of algorithms were proposed in the IR community, based on the principle of counting the number of appearances of each word in the text, assuming that the words with the highest number of appearances serve as the context. Variations on this simple mechanism involve methods for identifying the relevance of words to a domain, using methods such as stop-lists and inverse document frequency. For illustration purposes, we next provide a description of a context recognition algorithm that uses the Internet as a knowledge base to extract multiple contexts of a given situation, based on the streaming in text format of information that represents situations (Segev, Leshno, & Zviran 2004). This algorithm has been used in identifying

context of chat discussions and medical documents, and is currently part of the QUALEG solution.

Let $\mathcal{D} = \{P_1, P_2, ..., P_m\}$ be a series of textual descriptors representing a document, where for all P_i there exists a collection of sets of contexts $\mathcal{C}_{ij}$ so that for each i, $ist(\mathcal{C}_{ij}, P_i)$ for all j. That is, the textual proposition P_i is true in each of the set of contexts $\mathcal{C}_{ij}$. The granularity of the descriptors varies, based on the case at hand, and may be a single sentence, a single paragraph, a statement made by a single participant (in a chat discussion or a Shakespearian play), *etc*. The context recognition algorithm identifies the outer context set $\mathcal{C}$ defined by

$$ist(\mathcal{C}, \bigcap_{i=1}^{m} ist(\mathcal{C}_{ij}, P_i))\forall j.$$

The input to the algorithm is a stream, in text format, of information. The context recognition algorithm output is a set of contexts that attempts to describe the current scenario most accurately. The set of contexts is a list of words or phrases, each describing an aspect of the scenario. The algorithm attempts to reach results similar to those achieved by the human process of determining the set of contexts that describe the current scenario.

The context recognition algorithm consists of four major phases: collecting data, selecting contexts for each text, ranking the contexts, and declaring the current contexts. The phase of data collection includes parsing the text and checking it against a stop-list. To improve this process, the text can be checked against a domain-specific dictionary. The result is a list of keywords obtained from the text. The selection of the current context is based on searching the Internet for relevant documents according to these keywords and on clustering the results into possible contexts. The output of the ranking stage is the current context or a set of highest ranking contexts. The set of preliminary contexts that has the top number of references, both in number of Internet pages and in number of appearances in all the texts, is declared to be the current context. The success of the algorithm depends, to a great extent, on the number of documents retrieved from the Internet. With more relevant documents, less preprocessing (using methods such as Natural Language Processing) is needed in the data collection phase.

From an Automatically Extracted Context to Ontology Concepts

Given the uncertainty involved in automatically extracting contexts, sticking with a strict approach according to which a context belongs to an ontology concept only if it is an element in its associated point of accumulation, may be too restrictive. To illustrate this argument, Let $\mathcal{C}$ be a context in a point of accumulation x and let $\mathcal{C}'$ be an automatically extracted context. The following three scenarios are possible:

- $\mathcal{C} \subset \mathcal{C}'$**:** In this case the context extraction algorithm has identified irrelevant descriptors to be part of the context (false positives). Unless the set of descriptors in $\mathcal{C}'$ that are not in $\mathcal{C}$ is a context in x as well, $\mathcal{C}'$ will not be matched correctly.
- $\mathcal{C}' \subset \mathcal{C}$**:** In this case the context extraction algorithm has failed to identify some descriptors as relevant (false negatives). Therefore, $\mathcal{C}'$ will only be matched correctly if $\mathcal{C}$ is a context in the same filter base.
- $\mathcal{C} \nsubseteq \mathcal{C}' \wedge \mathcal{C}' \nsubseteq \mathcal{C}$**:** This is the case in which both false positives and false negatives exist in $\mathcal{C}'$.

A good algorithm for context extraction generates contexts in which false negatives and false positives are considered to be the exception, rather than the rule. Therefore, we would like to measure some "distance" between an extracted context and various points of accumulation, assuming a "closer" ontology concept to be better matched. To that end, we define a metric function for measuring the distance between a context and ontology concepts, as follows.

We first define distance between two descriptors $\langle c_i, w_i\rangle$ and $\langle c_j, w_j\rangle$ to be:

$$d(c_i, c_j) = \begin{cases} |w_i - w_j| & i = j \\ \max(w_i, w_j) & i \neq j \end{cases}$$

This distance function assigns greater importance to descriptors with larger weights, assuming that weights reflect the importance of a descriptor within a context. To define the best ranking concept in comparison with a given context we use Hausdorff metric. Let A and B be two contexts and a and b be descriptors in A and B, respectively. Then,

$$\begin{aligned} d(a, B) &= \inf\{d(a, b)|b \in B\} \\ d(A, B) &= \max\{\sup\{d(a, B)|a \in A\}, \sup\{d(b, A)|b \in B\}\} \end{aligned}$$

The first equation provides the value of minimal distance of an element from all elements in a set. The second equation identifies the furthest elements when comparing both sets.

Example 2. *Going back to our case study example, the context* $\{\langle$*Musik,* $8\rangle\}$, $\{\langle$*Open Air,* $1\rangle\}$ *may be relevant to both Perspective du Theatre and Long Day School, since in both, a descriptor Musik is found, albeit with different weights. The distance between* $\langle$*Musik,* $8\rangle$ *and* $\langle$*Musik,* $6\rangle$ *in Perspective du Theatre is* 2*, and to* $\langle$*Musik,* $2\rangle$ *in Long Day School is* 6*. Assume that* $\{\langle$*Open Air,* $1\rangle\}$ *is a false positive, which does not appear in neither Perspective du Theatre nor in Long Day School. Therefore, its distance from each of the two points accumulation is* 1 *(since* $\inf\{d(a, b)|b \in B\} = 1$*,* e.g.*, when comparing* $\{\langle$*Open Air,* $1\rangle\}$ *with* $\{\langle$*Kulturpolitik, 1*$\rangle\}$*). We can therefore conclude that the distance between the context and Perspective du Theatre is* 2*, which is smaller than its distance from Long Day School (computed to be* 6*). Therefore, Perspective du Theatre will be ranked higher than Long Day School.*

We defer to an extended version of this paper the design of efficient data structures to ensure efficient ranking computation. We now discuss the application of the ranking scheme to the three tasks of email routing, opinion analysis, and public agenda.

Email routing: The user provides QUALEG with a distance threshold t_1. Any ontology concept that matches with a context, automatically generated from an email,

and its distance is lower than the threshold ($d(A, B) < t_1$) will be considered relevant, and the email will be routed accordingly.

Opinion analysis: relevant set of ontology concepts are identified, similarly to email routing. Then for each ontology concept, the relative distance of the different opinions of that concept are evaluated. If the difference in distance is too close to call (given an additional threshold t_2), the system refrains from providing an opinion (and the email is routed accordingly). Otherwise, the email is marked with the opinion with minimal distance.

Public agenda: If all ontology concepts (of the n relevant concepts) satisfy that $d(A, B) \geq t_1$, the email is considered to be part of a new topic on the public agenda, and is added to other emails under this concept. Periodically, such emails are clustered and provided to decision makers to determine the addition of new ontology concepts.

Discussion and Conclusion

The paper presents a topological framework for combining contexts and ontologies in a model that maps contexts to ontologies. Contexts, individual views of a domain of interest, are matched to ontology concepts, often considered to be the "golden standard," for various purposes such as routing and opinion analysis. The model provides a conceptual structure, based on topological definitions, which delineates how and when contexts can be mapped to ontologies. The uncertainty, inherent to automatic context extraction, is managed through the definition of distance among contexts and a ranking of ontology concepts with respect to a given context.

To analyze the context and the mapping of contexts to ontologies, data from a local government, in the form of email messages from citizens, is used. The object of the local government is to analyze the quantities of information flowing in that could not be handled using its human resources. The information is examined to see whether the correct context could be identified and mapped to the right ontology. Since the project involves different countries and different languages, a multilingual ontology system is used. According to the model, different sets of words, representing the same concept, can be mapped to the multilingual ontology.

Each ontology concept was divided into positive and negative citizen opinions about the topics discussed in the email messages. This classification allows the local government to make decisions according to the citizen opinions, which are derived from the information received by email and analyzed only by the algorithm and not by a civil servant.

Initial experiments has yielded reasonable results. The results show that it is possible to automatically perform operations such as information routing and opinion analysis, based on the mapping of contexts and ontologies. We shall briefly provide here a few observations, gathered from the experiments. We defer a complete report on our experiments to an extended version of this paper.

During our experiments with the model we have identified several factors that may contribute to uncertainty. The main reason for errors in ontology concept identification pertains to the preprocessing of the input. The preprocessing was limited to a minimal and naïve dissection of the input. Most of the emails consisted of few sentences only, resulting in a one-shot attempt to determine the correct context. These results could be improved using different preprocessing methods, and the utilization of "soft" NLP tools. The ontology definition, which is currently restricted to a small number of words, also contributed to a low recall rate.

Some problems identified in the mapping of the context to ontology concepts were based on word association. For example, after an email ontology was identified as Perspectives du Theater, an attempt was made to identify its opinion. The number of positive words in the email were counted, and the result was three positive words taken from a predefined list. Therefore, the algorithm identified the opinion as positive. However, a single negative word in the email, not located on the list, transformed the opinion into a negative one. We are currently seeking more advanced techniques to improve opinion analysis. These methods include the analysis of the position of negative and positive words in an email.

As a final comment, we note that the current model assumes the availability of a predefined ontology. Therefore, ontology concepts and their relationships are provided beforehand, and newly extracted contexts are mapped to existing concepts. A possible direction for further research would be to utilize the partial overlapping among contexts to identify ontological relationships, such as generalization-specialization relationships.

Acknowledgments

The work of Gal was partially supported by two European Commission 6^{th} Framework IST projects, QUALEG and TerreGov, and the Fund for the Promotion of Research at the Technion. The authors thank Giora Dula for his useful comments. We thank Amir Taller for his assistance in integrating the Knowledge Extraction component with QUALEG infrastructure.

References

Assadi, H. 1998. Construction of a regional ontology from text and its use within a documentary system. *In Proceedings of the International Conference on Formal Ontology and Information Systems (FOIS-98)*.

Berge, C. 1997. *Topological Spaces*. Dover Publications.

Borgida, A., and Brachman, R. J. 1993. Loading data into description reasoners. In *Proceedings of the 1993 ACM SIGMOD international conference on Management of data*, 217–226.

Bunge, M. 1977. *Treatise on Basic Philosophy: Vol. 3: Ontology I: The Furniture of the World*. New York, NY: D. Reidel Publishing Co., Inc.

Bunge, M. 1979. *Treatise on Basic Philosophy: Vol. 4: Ontology II: A World of Systems*. New York, NY: D. Reidel Publishing Co., Inc.

Choset, H., and Nagatani, K. 2001. Topological simultaneous localization and mapping (slam): Toward exact local-

ization without explicit localization. *IEEE Trans. on Robotics and Automation* 17(2):125–137.

Chung, C. Y.; Lieu, R.; Liu, J.; Luk, A.; Mao, J.; and Raghavan, P. 2002. Thematic mapping from unstructured documents to taxonomies. In *Proceedings of the 11th International Conference on Information and Knowledge Management (CIKM)*.

Doan, A.; Madhavan, J.; Domingos, P.; and Halevy, A. 2002. Learning to map between ontologies on the semantic web. In *Proceedings of the eleventh international conference on World Wide Web*, 662–673. ACM Press.

Donini, F.; Lenzerini, M.; Nardi, D.; and Schaerf, A. 1996. Reasoning in description logic. In Brewka, G., ed., *Principles on Knowledge Representation, Studies in Logic, Languages and Information*. CSLI Publications. 193–238.

Gal, A.; Anaby-Tavor, A.; Trombetta, A.; and Montesi, D. 2005a. A framework for modeling and evaluating automatic semantic reconciliation. *VLDB Journal* 14(1):50–67.

Gal, A.; Modica, G.; Jamil, H.; and Eyal, A. 2005b. Automatic ontology matching using application semantics. *AI Magazine* 26(1).

H. Davulcu, S. V., and Nagarajan, S. 2003. Ontominer: Bootstrapping and populating ontologies from domain specific websites. In *Proceedings of the First International Workshop on Semantic Web and Databases*.

Kashyap, V., and Sheth, A. 1996. Semantic and schematic similarities between database objects: a context-based approach. *VLDB Journal* 5:276–304.

Kashyap, V.; Ramakrishnan, C.; Thomas, C.; and Sheth, A. 2005. Taxaminer: An experimentation framework for automated taxonomy bootstrapping. *International Journal of Web and Grid Services, Special Issue on Semantic Web and Mining Reasoning*. to appear.

Kifer, M.; Lausen, G.; and Wu, J. 1995. Logical foundation of object-oriented and frame-based languages. *Journal of the ACM* 42.

Koenig, S., and Simmons, R. 1996. Passive distance learning for robot navigation. In *Proceedings of the Thirteenth International Conference on Machine Learning (ICML)*, 266–274.

Madhavan, J.; Bernstein, P.; and Rahm, E. 2001. Generic schema matching with Cupid. In *Proceedings of the International conference on very Large Data Bases (VLDB)*, 49–58.

Madhavan, J.; Bernstein, P.; Domingos, P.; and Halevy, A. 2002. Representing and reasoning about mappings between domain models. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence and Fourteenth Conference on Innovative Applications of Artificial Intelligence (AAAI/IAAI)*, 80–86.

Maedche, A., and Staab, S. 2001. Ontology learning for the semantic web. *IEEE Intelligent Systems* 16.

McCarthy, J. 1993. Notes on formalizing context. *In Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*.

McGuinness, D.; Fikes, R.; Rice, J.; and Wilder, S. 2000. An environment for merging and testing large ontologies. In *Proceedings of the Seventh International Conference on Principles of Knowledge Representation and Reasoning (KR2000)*.

Melnik, S., ed. 2004. *Generic Model Management: Concepts and Algorithms*. Springer-Verlag.

Mena, E.; Kashyap, V.; Illarramendi, A.; and Sheth, A. P. 2000. Imprecise answers in distributed environments: Estimation of information loss for multi-ontology based query processing. *International Journal of Cooperative Information Systems* 9(4):403–425.

Motro, A., and Rakov, I. 1998. Estimating the quality of databases. *Lecture Notes in Computer Science*.

Noy, F. N., and Musen, M. 2000. PROMPT: Algorithm and tool for automated ontology merging and alignment. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI-2000)*, 450–455.

Papatheodorou, C.; Vassiliou, A.; and Simon, B. 2002. Discovery of ontologies for learning resources using word-based clustering. *Proceedings of the World Conference on Educational Multimedia, Hypermedia and Telecommunications (ED-MEDIA 2002)* 1523–1528.

Remolina, E., and Kuipers, B. 2004. Towards a general theory of topological maps. *Artificial Intelligence* 152:47–104.

Segev, A.; Leshno, M.; and Zviran, M. 2004. Context recognition using internet as a knowledge base. Technical Report TR-04-ISE-1, Technion.

Shatkay, H., and Kaelbling, L. 1997. Learning topological maps with weak local odometry information. In *Proc. IJCAI-97*.

Siegel, M., and Madnick, S. E. 1991. A metadata approach to resolving semantic conflicts. In *Proceedings of the 17th International Conference on Very Large Data Bases*, 133–145.

Simhon, S., and Dudek, G. 1998. A global topological map formed by local metric maps. *In IEEE/RSJ International Conference on Intelligent Robotic Systems* 3:1708–1714.

Spyns, P.; Meersman, R.; and Jarrar, M. 2002. Data modelling versus ontology engineering. *ACM SIGMOD Record* 31(4).

Terziyan, V., and Puuronen, S. 2000. Reasoning with multilevel contexts in semantic metanetwork. In P. Bonzon, M. Cavalcanti, R. N., ed., *Formal Aspects in Context*, 107–126. Kluwer Academic Publishers.

van Rijsbergen, C. J. 1979. *Information Retrieval*. London: Butterworths, second edition edition.

Vickery, B. 1966. *Faceted classification schemes*. New Brunswick, N.J.: Graduate School of Library Service, Rutgers, the State University.

Context-driven Disambiguation in Ontology Elicitation *

Pieter De Leenheer and **Aldo de Moor**
Semantics Technology and Applications Research Laboratory (STARLab)
Department of Computer Science
Vrije Universiteit Brussel
Pleinlaan 2, B-1050 BRUSSEL, Belgium
{pieter.de.leenheer,aldo.de.moor}@vub.ac.be

Abstract

Ontologies represent rich semantics in a lexical way. Lexical labels are used to identify concepts and relationships, though there is no bijective mapping between them. Phenomenons such as synonyms and homonyms exemplify this, and can result in frustrating misunderstanding and ambiguity. In the elicitation and application of ontologies, the meaning of the ontological knowledge is dependent on the context. We consider the role of context in ontology elicitation by introducing context in a concept definition server for ontology representation. We also adopt other features of context found in literature, such as packaging of knowledge, aligning elements of different contexts, and reasoning about contexts. Finally, we illustrate context-driven ontology elicitation with a real world case study.

Introduction

Though a vast amount of research has been conducted on formalising and applying knowledge representation (KR) models (Gruber 1993; Guarino 1998; Meersman 1999; Ushold & Gruninger 1996; Farquhar, Fikes, & Rice 1997), there is still a major problem with lexical disambiguation and subjectivity during the *elicitation* and *application* of an ontology. The problem is principally caused by two facts: (i) no matter how expressive ontologies might be, they are all in fact lexical representations of concepts, relationships, and semantic constraints; and (ii) linguistically, there is no bijective mapping between a concept and its lexical representation.

During the *elicitation* of an ontology (cfr. Fig. 1), its basic knowledge elements (such as concepts and relationships) are extracted from various resources such as (structured) documents and human domain experts. Many ontology approaches focus on the conceptual modelling task, hence the distinction between lexical level (term for a concept) and conceptual level (the concept itself) is often weak or ignored. In order to represent concepts and relationships lexically, they usually are given a uniquely identifying term (or label). However, the context of the resource the ontology element was extracted from is not unimportant, as the meaning of a concept behind a lexical term is influenced by this *context of elicitation*. Phenomenons such as synonyms and homonyms are typical examples of this, and can result in frustrating misunderstanding and ambiguity when unifying information from multiple sources. Similar for the *application* of an ontology: the interpretation of the knowledge elements (which are referred to by terms) of the ontology is ambiguous if the context of application, such as the purpose of the user, is not considered.

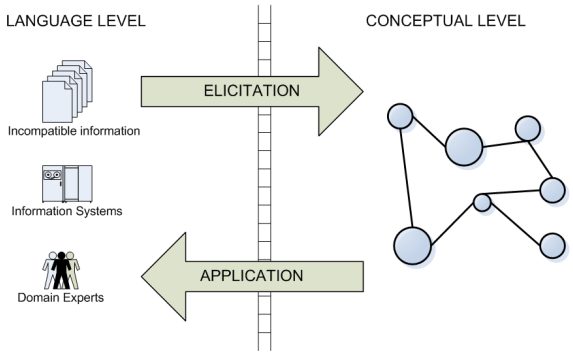


Figure 1: Ontologies are elicited by extracting knowledge from various sources and are also applied in different contexts.

Context was already introduced before to tackle lexical disambiguation (Buvač 1996b; Meersman 2001). In the literature, different, often unrelated interpretations of context in KR can be found. E.g., researchers introduced context to provide subjectivity: a context is a grouping of knowledge that provides a subjective (i.e. "context-dependent") view of a particular (community of) agent(s) (Guha & D. 1990; Lenat 1995; Theodorakis 1999). Our purpose in this paper, however, is to examine in detail the role that multiple contexts can play in the disambiguation of terms in the ontology elicitation process.

This paper is structured as follows: first, we give a synthesis of our literature study on context, and identify the fea-

*We would like to thank our colleagues in Brussels, especially Robert Meersman and Luk Vervenne, for the valuable discussions about theory and case. This research has been partially funded by the EU Leonardo da Vinci Project CODRIVE (BE/04/B/F/PP-144.339)

tures of contexts useful for our purpose. Then we present the DOGMA ontology representation framework, where we introduce the idea of context and a concept definition server to (i) logically group parts of knowledge, (ii) disambiguate the lexical representation of concepts and relationships, by distinguishing between language level and conceptual level, and (iii) build semantic bridges between different ontological contexts. We illustrate our framework by considering context-driven ontology elicitation in a real-world case study.

Contexts, Situations and Possible Worlds

Today in AI and linguistics, the word *context* has gained a (confusing) variety of meanings, which have led to diverse interpretations and purposes of context (Sowa 1995; 1997). Moreover, context is found in various AI application fields such as database integration (Farquhar *et al.* 1995), knowledge translation (Buvač & Fikes 1995), and reasoning (Giunchiglia 1993; Nayak 1994; McCarthy & Buvač 1994).

Peirce (Buchler 1955), with his preliminary notion of *sheets of assertion* was one of the pioneers in the formalisation of context. Although well known, there is no common understanding of its semantics. The theories of semantics based on *possible worlds* (Kripke 1963) or *multi-modal logic* are also associated with the notion of context. A proposition is assigned a true or false value depending on which world (read: "context") is considered among a set of possible worlds. Amongst this set of possible worlds there is the actual world, assigned w_0. Some worlds are accessible from some other worlds. A proposition is *necessarily* true in w_0 if it is true in every world accessible from w_0. A proposition is *possibly* true if it is true in some (at least one) possible world accessible from w_0. Instead of assuming possible worlds, Hintikka (1963) developed independently an equivalent semantics for multi-modal logic, which he called model sets.

McCarthy (1987; 1993) formalised context as first-class objects by introducing his basic relation $ist(\mathcal{C}, p)$. This ist predicate might be read as: "proposition p is true in context $\mathcal{C}$".

Situation semantics (Barwise & Perry 1983) is a reaction to multi-modal logic. While each possible worlds' model represents an large, open-ended, unbounded region of space-time, *situations* are smaller chunks that are more "manageable" (Sowa 2000, pp. 184).

Guha et al. (1990), adopt a notion of context for scaling the management of the very large knowledge base Cyc (Lenat 1995). They implement *microtheories* that allow assumptions in a limited context, but leave open the ability to use the knowledge in a larger context. Microtheories are organised in an subsumption hierarchy, where everything asserted in the super-microtheory, is true in the sub-microtheory, unless explicitly contradicted. Examples are theories of bibliography keeping, theories of car selling company, etc. Similar to McCarthy's conception, microtheories are inter-related via lifting rules stated in outer context.

McCarthy's intention to use context was predominantly for reasoning about relationships *between* contexts in an *outer context*. A proposition that is true in one context, might be asserted in another context under certain conditions. He calls this *lifting*. For example the same predicate p in context $\mathcal{C}_1$ can have a different name or arity, or be equivalent in another context $\mathcal{C}_2$. Generally, a lifting formula specifies an alignment between proposition and terms in subcontexts to possibly more general propositions and terms in an outer context. Subcontexts are often specialised with regard to time, place, and vocabulary. According to McCarthy however, there is no "root" context in which all axioms hold and everything is meaningful: for each axiom in a given context, one can always find a more general context in which the axiom fails.

McCarthy's work was further developed by Buvač and Guha (1991). Buvač (1996b) concluded that there is a need for common sense knowledge to lexically disambiguate a proposition like "Vanja is at a bank". From this proposition, the inference system cannot determine whether Vanja is at a river bank or at a financial bank. Note that McCarthy (1993) also concluded this as one of the reasons for introducing his formal context. He argued similarly that a term might have a particular meaning in a professional conversation different from the one in daily language use. This trend is reinforced by the field of linguistics (Langacker 1987).

Buvač (1996a) also extended the propositional logic of context to the quantificational case, providing a way to express first-order properties of contexts, and the truth of arbitrary predicates within a context.

Theodorakis (1999) introduces a context formalism from the perspective of data modeling. A context is an abstract object containing other objects, viz. data models. He organises different data models constructed from different perspectives or levels of abstraction into different contexts. These contexts are "integrated" by cross-referencing one to another. This is very similar to Guha's micro-theories.

In conceptual graph theory (Sowa 2000), context provides a means to describe what is true in a certain situation, without requiring the description of these situations per se (Mineau, Missaoui, & Godinx 2000). Mineau et al. (1997) propose to structure the world into a partial order of subsumption between contexts. The most general context (which does not exist in McCarthy's opinion) is called, in honour of Peirce, the *universal sheet of assertion*.

Giunchiglia (1993) was especially motivated by the problem of reasoning on a subset of the global knowledge base. The notion of context is used for this "localisation". His perspective is similar to (McCarthy 1987; 1993).

Guha (1991) defined a mapping from McCarthy's contexts to *situations* of Barwise and Perry (1983), i.e., for every context $\mathcal{C}$, there exists a situation s *and* for every proposition p: p is true in context $\mathcal{C}$ if and only if situation s is described by p. A similar parallel exists between logic of context and standard multi-modal logic. For more we refer to Halpern and Moses (1992).

Synthesis Based on our literature study, we distinguish four effective features of context (for reasoning), which we aim to integrate in our framework: (i) contexts package related knowledge: in that case a context defines (part of)

the knowledge of a particular domain; (ii) context provides pointer for lexical disambiguation; (iii) lifting rules provide an alignment between assertions in disconnected knowledge bases (or contexts); and (iv) "Statements about contexts are themselves in contexts" (McCarthy 1996); in other words, contexts can be embedded.

Summarising, to disambiguate terms, in general, an analysis of multiple contexts is needed. However, to implement these theoretical notions in real world systems is not trivial. Such implementation is the focus of the DOGMA ontology framework.

DOGMA Ontology Framework

DOGMA[1] is an ontology representation model and framework that separates the specification of the *conceptualisation* (i.e. lexical representation of concepts and their interrelationships) from its *axiomatisation* (i.e. semantic constraints). This principle corresponds to an orthodox *model-theoretic* approach to ontology representation and development. Consequently, the DOGMA framework consists of two layers: a *Lexon Base* and a *Commitment Layer*. A full formalisation of DOGMA Ontology is found in (De Leenheer & Meersman 2005).

Lexon Base

The Lexon Base is an uninterpreted, extensive and reusable pool of elementary building blocks for constructing an ontology. These building blocks (called *lexons*[2]) represent *plausible binary fact-types* (e.g., Person drives/is_driven_by Car). The Lexon Base is stored in an on-line DOGMA server. For guiding the ontology engineer through this very large database, *contexts* impose a meaningful grouping of these *lexons* within the Lexon Base. The context of a lexon refers to the source it was extracted from. Sources could be terminological[3] or human domain experts. A lexon is defined as:

Definition 1 *A lexon is an ordered 5-tuple of the form* $< \gamma, t_1, r_1, r_2, t_2 >$ *where* $\gamma \in \Gamma$, $t_1 \in T$, $t_2 \in T$, $r_1 \in R$ *and* $r_2 \in R$. Γ *is a set of identifiers,* T *and* R *are sets of strings in some alphabet* $\mathcal{A}$; t_1 *is called the headword of the lexon and* t_2 *is called the tailword of the lexon;* r_1 *is the role of the lexon,* r_2 *is the co-role;* γ *is the context in which the lexon holds.*

Role and co-role indicate that a lexon can be read in two directions. A lexon $< \gamma, t_1, r_1, r_2, t_2 >$ is a fact type that might hold in a domain, expressing that within the context γ, an object of type t_1 might plausibly play the role r_1 in relation to an object of type t_2. On the other hand, the same lexon states that within the same context γ, an object of type t_2 might play the co-role r_2 in (the same) relation to an object of type t_1.

Some role/co-role label pairs of lexons in the Lexon Base might intuitively express a specialisation relationship, e.g. $< \gamma, manager, is\,a, subsumes, person >$. However, as already mentioned above: the lexon base is uninterpreted, so the decision to interpret a role/co-role label pair as being a part-of or specialisation relation, is postponed to the commitment layer, where the semantic axiomatisation takes place.

Commitment Layer

Committing to the Lexon Base means selecting a meaningful set Σ of lexons from the Lexon Base that approximates well the intended[4] conceptualisation, and subsequently putting semantic constraints on this subset. The result (i.e., Σ plus a set of constraints), called an *ontological commitment*, is a logical theory that intends to model the meaning of this application domain. An ontological commitment constitutes an axiomatisation in terms of a network of lexons logically connected and provides a partial view of the Lexon Base. These networks are visualised in a NIAM[5]-like schema (cfr. Fig. 2). An important difference with the underlying Lexon Base is that commitments are internally unambiguous and semantically consistent[6]. Once elicited, ontological commitments (i.e. ontologies) are used by various applications such as information integration and mediation of heterogeneous sources. Though ontologies can differ in structure and semantics, they all are build on a shared Lexon Base.

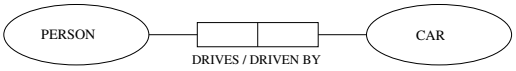


Figure 2: Illustration of a lexon that is described in a hypothetical context γ .

The commitments are specified in a designated language, called Ω-RIDL (Verheyden, De Bo, & Meersman 2004). It describes semantic constraints in terms of lexons, covering all classical database constraints (cfr. ORM). It also specifies which role/co-role label pairs are interpreted as which ontological relationship (such as subsumption, part-of). Consequently, this impacts the semantics of the commitment.

Commitments are also categorised and stored in a *commitment library* in the DOGMA server.

Contexts and Term Disambiguation

A lexon is a lexical representation of a conceptual relationship between two concepts, however, there is no bijective mapping between a lexical representation and a concept. Consider for example phenomenons such as synonyms and

[1] acronym for Developing Ontology-Guided Mediation of Agents; a research initiative of VUB STARLab

[2] Lexons are DOGMA knowledge elements.

[3] "A context refers to text, information in the text, to the thing the information is about, or the possible uses of the text, the information in it or the thing itself" (Sowa 2000, pp. 178).

[4] With respect to the application domain.

[5] NIAM (Verheijen & Van Bekkum 1982) is the predecessor of ORM (Halpin 2001).

[6] Although it is outside the scope of this paper, we find it valuable to note that in the research community it is debated that consistency is not necessarily a requirement for an ontology to be useful.

homonyms that can result in frustrating misunderstanding and ambiguity (see Def. 5). As we have seen, the meaning of a lexical term can vary depending on the context that holds.

In DOGMA, a context is used to group lexons that are related[7] to each other in the conceptualisation of a domain.

A context in DOGMA has one fundamental property: it is also a mapping function used to disambiguate terms by making them language-neutral. Based on Meersman (2001), we can give the following definition for a context:

Definition 2 *A context $\gamma \in \Gamma$ is a mapping $\gamma : T \cup R \rightarrow C$ from the set of terms and roles to the set of concept identifiers in the Universe of Discourse (UoD) C. In a context, every term or role is mapped to at most one concept identifier. A context γ is also a reference to one or more documents and/or parts of a document[8]. This reference is defined by the mapping $cd : \Gamma \rightarrow \mathcal{D}$.*

In this case we can check which lexons are valid in that specific context, more specifically those lexons extracted from (the parts of) the documents to which the context γ refers. A tuple $< \gamma, t >$ identifies a unique concept. With a concept we mean the thing itself to which we refer by means of a term (or role) in the Lexon Base. If we want to describe the set of concepts of our UoD formally, we can do this, according to Meersman (2001), by introducing the partial function $ct : \Gamma \times T \cup R \rightarrow C$ which associates a concept with a tuple consisting of a context and a term (or role). This partial function, which describes a form of *meaning articulation*, is defined as follows:

Definition 3 (meaning articulation) *Given the partial function $ct : \Gamma \times T \cup R \rightarrow C$, then*

$$ct(\gamma, t) = c \Leftrightarrow \gamma(t) = c.$$

An association $ct(\gamma, t) = c$ is called the "meaning articulation" or articulation[9] of a term t (in a particular context γ) into a concept identifier c. ct is called a meaning articulation mapping.

The set of concept identifiers C of the UoD can be formally defined as:

Definition 4 *The set of concepts identifiers $C = \{ct(\gamma, t) | \gamma \in \Gamma, t \in T \cup R\}$.*

Example 1 illustrates the two latter definitions:

Example 1 *Consider a term "capital". If this term was elicited from a typewriter manual, it has a different meaning than when elicited from a book on marketing. Hence, we have resp. two contexts: γ_1 = typewriter manual, and γ_2 = marketing book. To express that "capital" is associated with different meanings, we write $ct(\gamma_1, capital) = c_1$, and $ct(\gamma_2, capital) = c_2$.*

[7] Not necessarily in a logical way but more in an informal way. E.g., lexons are related because they were elicited from the same source, i.e. the elicitation context.

[8] At this stage, we only require a document should provide information what or whom the lexon was elicited from. See our case study below for a concrete example.

[9] We adopt the term articulation from Mitra et al. (2000) (see discussion).

Until now, the endpoint of the meaning articulation is a meaningless concept identifier $c_1, c_2 \in C$. However, in the next section we will introduce the Concept Definition Server. Each concept identifier itself will point to a particular concept definition. The terms (on the *language level*) that are articulated (using ct) are then mapped to a particular *explication* of a meaning, i.e. a concept definition of a term residing in the Concept Definition Server (on the *conceptual level*), instead of to a meaningless concept identifier.

Before we continue, some useful terminology, as defined by De Bo and Spyns (2004), is presented in Def. 5:

Definition 5

- *Two terms $t_1 \in T$ and $t_2 \in T$ are synonyms within a context γ if and only if $(\gamma(t_1) = c \Leftrightarrow \gamma(t_2) = c)$.*
- *Two identical terms $t \in T$ are called homonyms if and only if $\exists \gamma_1, \gamma_2 \in \Gamma : \gamma_1(t) \neq \gamma_2(t)$.*

These definitions also hold for roles $r \in R$.

Completing the Articulation: Concept Definition Server

The idea for a Concept Definition Server (CDS) was first mentioned in (De Bo, Spyns, & Meersman 2004), and is based on the structure of Wordnet (Fellbaum 1998). CDS is a database in which you can query for a term, and get a set of different meanings or *concept definitions* (called *senses* in Wordnet) for that term. A concept definition is unambiguously explicated by a gloss (i.e. a natural language (NL) description) and a set of synonymous terms. Consequently we identify each concept definition in the CDS with a concept identifier $c \in C$.
The following definition specifies the CDS:

Definition 6 *We define a Concept Definition Server Υ as a triple $< T_\Upsilon, \mathcal{D}_\Upsilon, concept >$ where:*

- *T_Υ is a non-empty finite set of strings (terms)[10];*
- *$\mathcal{D}_\Upsilon$ is a non-empty finite document corpus;*
- *$concept : C \longmapsto \mathcal{D}_\Upsilon \times \wp(T_\Upsilon)$ is an injective mapping between concept identifiers $c \in C$ and concept definitions.*

Further, we define $conceptdef(t)$

$$= \{concept(c) \,|\, concept(c) =< g, sy > \wedge t \in sy\},$$

where gloss $g \in \mathcal{D}_\Upsilon$ and synset $sy \subseteq T_\Upsilon$.

Going from the language level to the conceptual level corresponds to articulating lexons into meta-lexons:

Definition 7 *Given a lexon $l :=< \gamma, t_1, r_1, r_2, t_2 >$, and an instance of an articulation mapping $ct : \Gamma \times T \cup R \rightarrow C$ with $ct(\gamma, t_1) = c_{t_1}, ct(\gamma, r_1) = c_{r_1}, ct(\gamma, r_2) = c_{r_2}, ct(\gamma, t_2) = c_{t_2}$ ($c_{t_1}, c_{r_1}, c_{r_2}, c_{t_2} \in C$). A meta-lexon $m_{l,ct} :=< c_{t_1}, c_{r_1}, c_{r_2}, c_{t_2} >$ (on the conceptual level) is the result of "articulating" lexon l via ct.*

[10] Additionally, we could require $T \cup R \subseteq T_\Upsilon$ (T and R from the Lexon Base). Doing so, we require each term and role in the Lexon Base to be a term in the synset of at least one concept definition.

In Fig. 3 the articulation is illustrated by a *meaning ladder* going from the (lower) language level to the (higher) conceptual level and vice-versa. This ladder is inspired by Stamper's *semiotic ladder*. Stamper (1973) argues that it is naive to see information as a primitive or atomic concept. From his operational point of view he means that in defining something, it is important to specify precisely by what procedure or operations to measure or perform. Hence, the solution in attempting to define "information" is to see information as signs and to define the different aspects or levels of these signs based on the operations one can do on these signs. His semiotic ladder consists of six views on signs (levels) from the perspective of physics, empirics, syntactics, semantics, pragmatics and the social world, that together form a complex conceptual structure. We refer to Fig. 1, where we introduced the levels and the ladder in the application–elicitation setting.

Next, another interesting definition can be given:

Definition 8 (Meta-lexon Base) *Given a Lexon Base Ω and a total articulation mapping $ct : \Gamma \times T \cup R \rightarrow C$, a Meta-lexon Base $M_{\Omega,ct} = \{m_{l,ct} | l \in \Omega\}$ can be induced.*

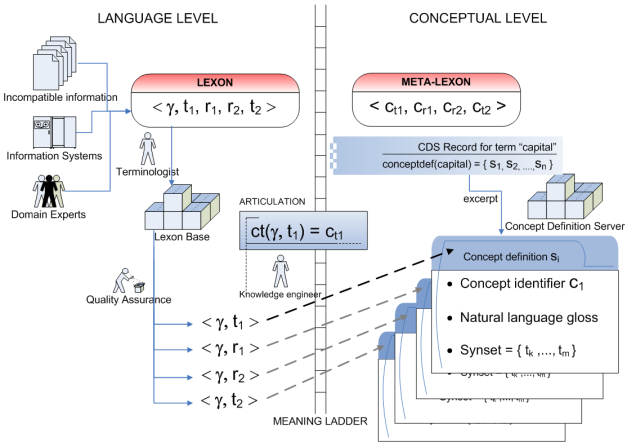


Figure 3: Illustration of the two levels in DOGMA ontology: on the left – the lexical level, lexons are elicited from various contexts. On the right, there is the conceptual level consisting of a concept definition server. The meaning ladder in between illustrates the articulation of lexical terms into concept definitions.

Example 2 *As an illustration of the defined concepts, consider Fig. 4. The term "capital" in two different contexts can be articulated to different concept definitions in the CDS. The terms are part of some lexons residing in the Lexon Base. The knowledge engineer first queries the CDS Υ for the various concept definitions of the term: $conceptdef(capital) = S_{capital} \subseteq \mathcal{D}_\Upsilon \times \wp(T_\Upsilon)$. Next, he articulates each term to the concept identifier of the appropriate concept definition:*

- *Term "capital" was extracted from a typewriter manual, and is articulated to concept identifier c_1 that corresponds to concept definition (or meaning) $s_1 \in S_{capital}$ (as illustrated on the right of Fig. 4). A gloss and set of synonyms (synset) is specified for s_1:*

$$concept\big(ct(typewriter\ manual, capital)\big) = s_1.$$

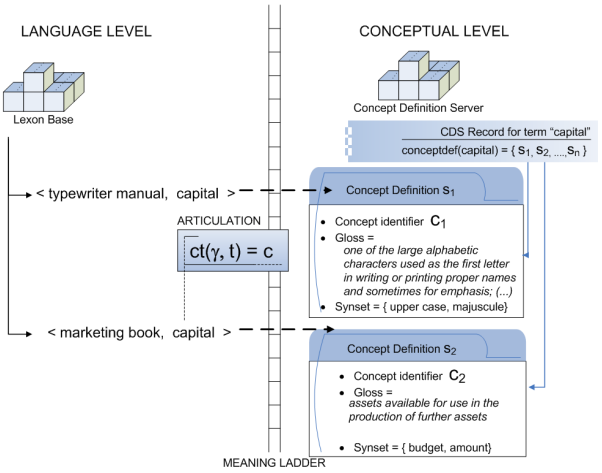


Figure 4: Illustration of two terms (within their resp. contexts), being articulated (via the mapping ct) to their appropriate concept definition.

- *Term "capital" was extracted from a marketing book, due to the different context it was extracted from, it is articulated to another concept identifier c_2 that is associated with a concept definition $s_2 \in S$:*

$$concept\big(ct(marketing\ book, capital)\big) = s_2.$$

On the other hand, suppose we have elicited a term "exercise" from the typewriter manual, and a term "example" from the marketing book. The engineers decide independently to articulate the resp. terms to the same concept definition with concept identifier c_3 with gloss: "a task performed or problem solved in order to develop skill or understanding":

$$\begin{aligned} c_3 &= ct(typewriter\ manual, exercise) \\ &= ct(marketing\ book, example). \end{aligned}$$

This articulation defines a semantic bridge between two terms in two different ontological contexts.

Shared Competency Ontology-Building

In this section we illustrate context-driven ontology elicitation in a realistic case study of the European CODRIVE[11] project.

Competencies and Employment

Competencies describe the skills and knowledge individuals should have in order to be fit for particular jobs. Especially in the domain of vocational education, having a central shared and commonly used competency model is becoming crucial in order to achieve the necessary level of

[11] CODRIVE is an EU Leonardo da Vinci Project (BE/04/B/F/PP-144.339).

interoperability and exchange of information, and in order to integrate and align the existing information systems of competency stakeholders like schools or public employment agencies. None of these organisations however, have successfully implemented a company-wide "competency initiative", let alone a strategy for inter-organisational exchange of competency related information.

The CODRIVE project aims at contributing to a competency-driven vocational education by using state-of-the-art ontology methodology and infrastructure in order to develop a conceptual, shared and formal KR of competence domains. Domain partners include educational institutes and public employment organisations from various European countries. The resulting shared "Vocational Competency Ontology" will be used by all partners to build interoperable competency models.

In building the shared ontology, the individual ontologies of the various partners need to be aligned *insofar necessary*. It is important to realise that costly alignment efforts only should be made when necessary for the shared collaboration purpose. In order to effectively and efficiently define shared relevant ontological meanings, context is indispensable.

Example: Adding a Term

The example concerns two participating organisations EI and PE, instances of resp. *educational institute* and *public employment agency*. Core shared and individual ontologies have already been defined for both EI and PE. Fig. 5 illustrates the different contexts[12] called resp. $SHARED$, EI and PE. The $SHARED$ ontology has amongst its concepts $Task$, with as subtypes $Educational\,Task$ and $Job\,Task$ (is_a represented by lexons). The concepts as referred in lexons are in fact terms, but within the context $SHARED$ they refer to at most one concept definition. The concepts underlined in the rules below are also modelled but not shown, similarly for the specialisation hierarchies of EI and PE.

EI is a national bakery institute, responsible for defining curriculum standards for bakery courses. It now wants to add a new term "panning" to the ontology. It defines this informally as the act of "depositing moulded dough pieces into baking pans with their seam facing down". Fig. 5 shows the steps in ontology alignment: step 1 is adding the new term (which resides in some lexon which is not shown) to ontology EI. Step 2 is the meaning articulation, illustrated by an arrow going from the language level of the ontology to a concept definition in the CDS.

Step 3 is triggered by the following informal rule:

R1: The CODRIVE ontology server (COS) asks EI to classify the shared concept to which the term belongs.

The EI representative classifies panning as an $Educational\,Task$ (illustrated as an arrow labelled with step 3). COS now looks in its ontological meta-model. One of the rules there demands:

R2: IF a NewTask is an EducationalTask, and the IndividualOntologyOwner is an EducationalInstitute THEN a Full semantic analysis of the NewTask needs to be added to the IndividualOntology of the IndividualOntologyOwner;

another meta-rule fires as an immediate consequence:

R3: IF a FullSemanticAnalysis needs to be made of a Concept in an IndividualOntology or SharedOntology THEN the ConceptTemplate needs to be filled out in that Ontology. Furthermore, for each Term and Role of that definition, a MeaningArticulation needs to be defined.

This means that in this case the template states it is necessary to know who is the performer of the task (e.g. $Student$), what inputs are necessary for the task (e.g. Pan, $Dough$), what is the result of the task ($Pan\,with\,Dough$), and so on. Rules **R2** and **R3** trigger step 4: in the EI context, the new task $Panning$ is semantically analysed, which boils down to extending the description in terms of lexons (illustrated by the lexons within the dashed box). Similarly to step 1, each new term or role in this box must be articulated (not shown in the figure).

Concurrently another rule triggers step 5:

R4: IF an EducationalTask is added to an IndividualOntology THEN a corresponding JobTask needs to be defined in all instances of IndividualOntology of all PublicEmploymentAgencies;

The rationale for this rule is that public employment agencies need to be aware of changes to the curricula of educational institutes, so that they are better able to match job seekers with industry demands. However, unlike the definitions of educational tasks, the job task definitions in

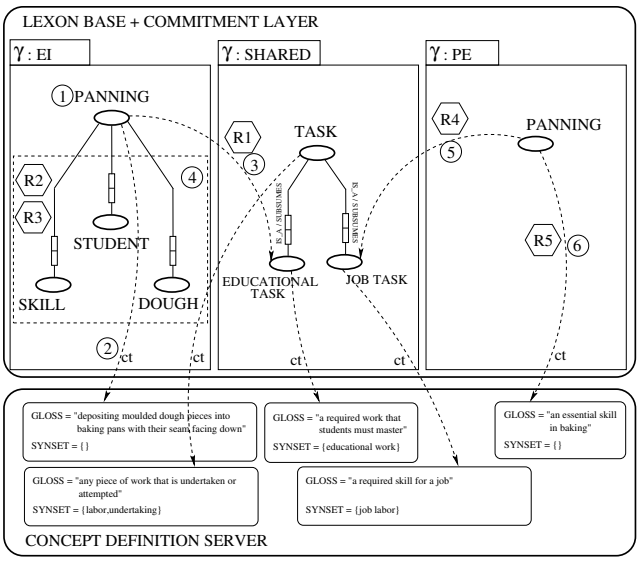


Figure 5: Illustration of the case study: top level, from right to left: ontologies EI, $SHARED$, and PE. On the bottom: CDS.

[12] In this case study, each context corresponds to exactly one ontology and vice-versa. However, an ontology engineer might select lexons from various contexts for modelling his ontology.

public employment agency ontologies only require a short informal description of the concept itself, not an extended template definition (step 6):

R5: IF a JobTask is added to an IndividualOntology THEN a Gloss needs to be defined for that Concept.

Of course, public employment agencies also could have the need for template definitions, but those would refer to the job matching processes in which the tasks play a role (*why* is panning needed), not to *how* the tasks themselves are to be performed.

Note the density of lexon elicitation in the EI ontology (on the left of Fig. 5) compared to the sparsely populated PE ontology (on the right of Fig. 5). The density reflects the minimal level of modelling details needed. Our context-driven ontology elicitation avoids wasting valuable modelling time and enormous cost.

This real world example illustrates one simple problem, viz. identical terms in different contexts can have different meanings (homonyms). However, disambiguation can become very complex when considering e.g., synonymy.

Discussion and Future Directions

Shamsfard et al. (2003) provide a comprehensive survey of methods and tools for (semi-)automatic ontology elicitation. However, in this paper our focus is not on automation. Work which is strongly related with what we need is e.g., Mitra et al. (2000), who indirectly adopt some of those features we concluded with in our synthesis earlier. They illustrate a semi-automatic tool for creating a mapping between two ontologies (in fact contexts). Their motivation is that two different terms can have the same meaning and the same term can have different meanings, which exactly defines the lexical disambiguation problem. This mapping is manifested by an *articulation ontology*, which is automatically generated from a set of *articulation rules* (i.e. semantic relationships) between concepts in each context resp.

In our framework, the CDS provides a basic means for relevant alignment of heterogeneous ontologies. The concept definitions (gloss, synset) in the CDS support the meaning articulation of language level terms. As was illustrated in Ex. 2, the articulation of terms from different contexts to a shared CDS, results in cross-context equivalence relations, i.e. synonyms.

The meta-rules we made in step 3 and 5, were not formalised explicitly in this paper. However, we could define a syntax, e.g.:

$$\begin{aligned} ct(EI, panning) &\preceq ct(SHARED, educational\,task) \\ ct(PE, panning) &\preceq ct(SHARED, job\,task) \end{aligned}$$

The semantics of the relation $\preceq$ is comparable to McCarthy's lifting rules: it allows us to specify an alignment between a term in one context to a possibly more general term in another context. In the future we will extend this feature and provide a formal semantics of meta-rules. This can be very powerful in context-driven ontology elicitation and application such as meta-reasoning on context and ontology alignment processes, and meaning negotiation processes between stakeholders. Currently we are exploring reasoning for commitment analysis and conceptual graph tools for ontology elicitation and analysis.

Initially, the lexon base and CDS are empty, but the CDS can be easily populated by importing similar information from publically available electronic lexical databases, such as Wordnet (Fellbaum 1998) or Cyc (OpenCyc). The Lexon Base is populated during the first step of an ontology elicitation process by various (not necessarily human) agents. See Reinberger & Spyns (2005) for unsupervised text mining of lexons. The second step in the elicitation process is to articulate the terms in the "learned" lexons.

Finally, we note that in the case study we did not consider the semantic constraints completely. The only visible constraint is the interpretation of the role/co-role pair is_a/subsumes as the ontological specialisation relation in ontology $SHARED$. Adding other relations such as defined in conceptual graph theory (Sowa 2000), will considerably improve the power of our meaning articulation approach.

Conclusion

We have presented an extension to the DOGMA ontology framework that enables context-driven ontology elicitation. We introduced contexts and a concept definition server to (i) logically group knowledge, (ii) to disambiguate the lexical representation of concepts and relationships, by distinguishing between language level and conceptual level, and (iii) to build semantic bridges between different ontological contexts. Next, we illustrated context-driven ontology elicitation considering a real world case example. Finally, we summarised related work and showed how our work can be extended.

References

Barwise, J., and Perry, J. 1983. *Situations and Attitudes*. MIT Press.

Buchler, J. 1955. *Philosophical Writings of Peirce*. New York: Dover Publ.

Buvač, S., and Fikes, R. 1995. A declarative formalization of knowledge translation. In *Proc. of 4th Int'l Conf. on Information and Knowledge Management (ACM CIKM 95)*.

Buvač, S. 1996a. Quantificational logic of context. *AAAI/IAAI* 1:600–606.

Buvač, S. 1996b. Resolving lexical ambiguity using a formal theory of context. In Van Deemter, K., and Peters, S., eds., *Semantic Ambiguity and Underspecification*. CSLI Publications.

De Bo, J.; Spyns, P.; and Meersman, R. 2004. Assisting ontology integration with existing thesauri. In *Proc. of On the Move to Meaningful Internet Systems (OTM2004) (Ayia Napa, Cyprus)*, 801–818. Springer Verlag.

De Leenheer, P., and Meersman, R. 2005. Towards a formal foundation of DOGMA ontology: part I. Technical Report STAR-2005-06, VUB STARLab.

Farquhar, A.; Dappert, A.; Fikes, R.; and Pratt, W. 1995. Integrating information sources using context logic. In Knoblock, C., and Levy, A., eds., *Information Gathering from Heterogeneous, Distributed Environments*.

Farquhar, A.; Fikes, R.; and Rice, J. 1997. The ontolingua server: a tool for collaborative ontology construction. *Int'l Journal of Human-computer Studies* 46(6):707–727.

Fellbaum, C., ed. 1998. *Wordnet, an Electronic Lexical Database*. MIT Press.

Giunchiglia, F. 1993. Contextual reasoning. *special issue on I Linguaggi e le Macchine* XVI:345–364.

Gruber, T. 1993. Cyc: a translation approach to portable ontologies. *Knowledge Acquisition* 5(2):199–220.

Guarino, N. 1998. Formal ontology and information systems. In *Proc. of the 1st Int'l Conf. on Formal Ontologies in Information Systems (FOIS98) (Trento, Italy)*, 3–15. IOS Press.

Guha, R., and D., L. 1990. Cyc: a midterm report. *AI Magazine* 11(3):32–59.

Guha, R. V. 1991. Contexts: a formalization and some applications. Technical Report STAN-CS-91-1399, Stanford Computer Science Department, Stanford, California.

Halpern, J., and Moses, Y. 1992. A guide to completeness and complexity for modal logics of knowledge and belief. *Artificial Intelligence* 54:319–379.

Halpin, T. 2001. *Information Modeling and Relational Databases (From Conceptual Analysis to Logical Design)*. Morgan Kauffman.

Hintikka, J. 1963. The modes of modality. In *Acta Philosophica Fennica, Modal and Many-valued Logics*, 65–81.

Kripke, S. 1963. Semantic analysis of modal logic i. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik* 9:67–96.

Langacker, R. 1987. *Foundations of Conditional Grammars*. Stanford University Press.

Lenat, D. 1995. Cyc: a large-scale investment in knowledge infrastructure. *Communications of the Association for Computing Machinery* 38(11).

McCarthy, J., and Buvač, S. 1994. Formalizing context (expanded notes). Technical Report STAN-CS-TN-94-13, Stanford University.

McCarthy, J. 1987. Generality in artificial intelligence. *Communication of the Association for Computing Machinery* 30(12):1030–1035.

McCarthy, J. 1993. Notes on formalizing context. In *Proc. of the 15th Int-l Joint Conf. Artificial Intelligence (IJCAI93) (Chambry, France)*, 555–560. Morgan Kaufmann.

McCarthy, J. 1996. A logical ai approach to context. Unpublished note.

Meersman, R. 1999. The use of lexicons and other computer-linguistic tools in semantics, design and cooperation of database systems. In *Proc.of the Conf. on Cooperative Database Systems (CODAS99)*, 1–14. Springer Verlag.

Meersman, R. 2001. Reusing certain database design principles, methods and techniques for ontology theory, construction and methodology. Technical report, VUB STAR Lab, Brussel.

Mineau, G., and Gerbé, O. 1997. Contexts: A formal definition of worlds of assertions. In *Proc.of the 5th Int'l Conf. on Conceptual Structures (ICCS-97)(Seattle, USA)*, 80–94. Springer Verlag.

Mineau, G.; Missaoui, R.; and Godinx, R. 2000. Conceptual modeling for data and knowledge management. *Data and Knowledge Engineering* 33(2):137–168.

Mitra, P.; Wiederhold, G.; and Kersten, M. L. 2000. A graph-oriented model for articulation of ontology interdependencies. In *EDBT '00: Proceedings of the 7th International Conference on Extending Database Technology*, 86–100. London, UK: Springer-Verlag.

Nayak, P. 1994. Representing multiple theories. In *Proc. of the 12th Nat'l Conf. on Artificial Intelligence (AAAI 94)(Seattle, Washington)*. AAAI Press.

OpenCyc. http://www.opencyc.org.

Reinberger, M.-L., and Spyns, P. 2005. Unsupervised text mining for the learning of DOGMA-inspired ontologies. In *Buitelaar P., Handschuh S., and Magnini B.,(eds.), Ontology Learning and Population*, in press. IOS Press.

Shamsfard, M., and Barforoush, A. A. 2003. The state of the art in ontology learning: a framework for comparison. *the Knowledge Engineering Review* 18(4):293–316.

Sowa, J. 1995. Syntax, semantics, and pragmatics of contexts. In *Proc. of the 3rd Int'l Conf. on Conceptual Structures: Applications, Implementation and Theory*, 1–15. Springer-Verlag.

Sowa, J. 1997. Peircean foundations for a theory of context. In *Conceptual Structures: Fulfilling Peirce's Dream*, 41–64. Springer-Verlag.

Sowa, J. 2000. *Knowledge Representation - Logical, Philosophical and Computational Foundations*. Brooks/Cole Publishing Co.

Stamper, R. 1973. *Information in Business and Administrative Systems*. John Wiley and Sons.

Theodorakis, M. 1999. *Contextualization: an Abstraction Mechanism for Information Modeling*. Ph.D. Dissertation, University of Crete, Greece.

Ushold, M., and Gruninger, M. 1996. Ontologies: Principles, methods and applications. *The Knowledge Engineering Review* 11(2):93–136.

Verheijen, G., and Van Bekkum, J. 1982. Niam, an information analysis method. In *Proc. of the IFIP TC-8 Conference on Comparative Review of Information System Methodologies (CRIS 82)*. North-Holland.

Verheyden, P.; De Bo, J.; and Meersman, R. 2004. Semantically unlocking database content through ontology-based mediation. In *Proc. of the 2nd Workshop on Semantic Web and Databases, VLDB Workshops (SWDB 2004) (Toronto, Canada)*, 109–126. Springer-Verlag.

Towards a Theory of Formal Classification

Fausto Giunchiglia and **Maurizio Marchese** and **Ilya Zaihrayeu**
{fausto, marchese, ilya}@dit.unitn.it
Department of Information and Communication Technology
University of Trento, Italy

Abstract

Classifications have been used for centuries with the goal of cataloguing and searching large sets of objects. In the early days it was mainly books; lately it has become Web pages, pictures and any kind of electronic information items. Classifications describe their contents using natural language labels, an approach which has proved very effective in manual classification. However natural language labels show their limitations when one tries to automate the process, as they make it almost impossible to reason about classifications and their contents. In this paper we introduce the novel notion of *Formal Classification*, as a graph structure where labels are written in a logical concept language. The main property of Formal Classifications is that each node can be associated a normal form formula which univocally describes its contents. This in turn allows us to reduce document classification and query answering to fully automatic propositional reasoning.

Introduction

In today's information society, as the amount of information grows larger, it becomes essential to develop efficient ways to summarize and navigate information from large, multivariate data sets. The field of classification supports these tasks, as it investigates how sets of "objects" can be summarized into a small number of classes, and it also provides methods to assist the search of such "objects" (Gordon Second edition 1999). In the past centuries, classification has been the domain of librarians and archivists. Lately a lot of interest has focused also on the management of the information present in the web: see for instance the WWW Virtual Library project[1], or the directories of search engines like Google, or Yahoo!.

Standard classification methodologies amount to manually organizing topics into hierarchies. Hierarchical library classification systems (such as the Dewey Decimal Classification System (DDC)[2] or the Library of Congress classification system (LCC)[3]) are attempts to develop static, hierarchical classification structures into which all of human knowledge can be classified. Although these are standard and universal techniques; they have a number of limitations:

- both classification and search tasks do not scale to large amounts of information. This is because, among other things, at any given level in such a hierarchy, there may be more than one choice of topic under which an object might be classified or searched.
- the semantics of a given topic is implicitly codified in a natural language label. These labels must therefore be interpreted and disambiguated.
- the semantic interpretation of a given topic depends also on the meanings associated to the labels at higher levels in the hierarchy (Magnini, Serafini, & Speranza 2003).

In the present paper we propose a formal approach to classification, capable of capturing the implicit knowledge present in classification hierarchies, and of supporting automated reasoning to help humans in their classification and search tasks. To this end, we propose a two step approach:

- first we convert a classification into a new structure, which we call *Formal Classification* (*FC*), where all the labels are expressed in a Propositional Description Logic language[4], that we call the Concept Language.
- then we further convert a FC into a *Normalized Formal Classification* (*NFC*). In NFCs each node is associated a Concept Language formula, that we call the *concept at a node*, which univocally codifies the node contents, taking into account both the label of the node and its position within the classification.

NFCs and concepts at nodes have many nice properties. Among them:

- they can be expressed in Conjunctive and/or Disjunctive Normal Forms (CNF / DNF). This allows humans and machines to easily inspect and reason on classifications (both visually and computationally).
- document classification and query answering can be done simply exploiting the univocally defined semantics codified in concepts at nodes. There is no need to inspect the edge structure of the classification.

[1] The WWW Virtual Library project, see http://vlib.org/.

[2] The Dewey Decimal Classification system, see http://www.oclc.org/dewey/.

[3] The Library of Congress Classification system, see http://www.loc.gov/catdir/cpso/lcco/lcco.html/.

[4] A Propositional Description Logic language is a Description Logic language (Baader *et al.* 2003) without roles.

- concepts of nodes are organized in a taxonomic structure where, from the root down to the leaves of the classification, child nodes are subsumed by their parent nodes.

The remainder of the paper is organized as follows. In the next section we introduce and present examples of standard classifications. We then introduce the definition of FC and discuss its properties. Afterwards, we introduce the notion of NFC and its properties. Next, we show how the two main operations performed on classifications, namely classification and search, can be fully automated in NFCs as a propositional satisfiability problem. The related and future work conclude the paper.

Classifications

Classifications are hierarchical structures used to organize large amounts of objects (Magnini, Serafini, & Speranza 2003). These objects can be of many different types, depending on the characteristics and uses of the classification itself. In a library, they are mainly books or journals; in a file system, they can be any kind of file (e.g., text files, images, applications); in the directories of Web portals, the objects are pointers to Web pages; in market places, catalogs organize either product data or service titles. Classifications are useful for both objects classification and retrieval. Users browse the hierarchies and quickly catalogue or access the objects associated with different concepts and linked to natural languages labels. We define the notion of Classification as follows:

Definition 1 (Classification) *A Classification is a rooted tree described by a triple $H = \langle C, E, l\rangle$ where C is a finite set of nodes, E is a set of edges on C, and l is a function from C to a set L of labels expressed in a natural language.*

In the rest of this section we describe and briefly discuss two different Classifications: a librarian classification hierarchy Dewey Decimal Classification system (DDC), and an example from a modern web catalogue, namely the Amazon book categories catalogue.

Example 1 (DDC). Since the 19^{th} century, librarians have used DDC to organize vast amounts of books. DDC divides knowledge into ten different broad subject areas, called classes, numbered 000 - 999. Materials which are too general to belong to a specific group (encyclopedias, newspapers, magazines, etc.) are placed in the 000's. The ten main classes are divided up into smaller classes by several sets of subclasses. Smaller divisions (to subdivide the topic even further) are created by expanding each subclass and adding decimals if necessary. A small part of the DDC system is shown on Figure 1.

In DDC, the notation (i.e., the system of symbols used to represent the classes in a classification system) provides a universal language to identify the class and related classes.

Before a book is placed on the shelves it is:

- classified according to the discipline matter it covers (given the Dewey number);
- some letters (usually three) are added to this number (usually they represent the author's last name);

```
500  Natural Science and Mathematics
520  Astronomy and allied sciences
   523  Specific celestial bodies and phenomena
      523.1  The universe
      523.2  Solar system
      523.3  The Earth
      523.4  The moon
      523.5  Planets
         523.51  Mercury
         523.52  Venus
         523.53  Mars          → 523.53HAN
         ...
```

Figure 1: A part of the DDC system with an example of book classification

- the number is used to identify the book and to indicate where the book will be shelved in the library. Books can be assigned a Dewey number corresponding to both leaf and non-leaf nodes of the classification hierarchy.

Since parts of DDC are arranged by discipline, not subject, a subject may appear in more than one class. For example, the subject "clothing" has aspects that fall under several disciplines. The psychological influence of clothing belongs in 155.95 as part of the discipline of psychology; customs associated with clothing belong in 391 as part of the discipline of customs; and clothing in the sense of fashion design belongs in 746.92 as part of the discipline of the arts. However, the final Dewey number associated to a book is unique and the classifier needs to impose a classification choice.

As an example, let's see how to determine the Dewey number for the following book: Michael Hanlon, "The Real Mars". A possible classification is Dewey number: 523.53 HAN and the classification choice for the book is shown in Figure 1.

The main properties of DDC are:

- the classification algorithm relies on the "Get Specific" criterion[5]: when you add a new object, get as specific as possible: dig deep into the classification schema, looking for the appropriate sub-category; it is bad practice to submit an object to a top level category, if one more specific exists. At present, the enforcement of such criterion is left to the experience of the classifier.
- each object is placed in exactly one place in the hierarchy. As a result of this restriction, a classifier often has to choose arbitrarily among several reasonable categories to assign the classification code for a new document (see the above example for "clothing"). Despite the use of documents called "subject authorities", which attempt to impose some control on terminology and classification criteria, there is no guarantee that two classifiers make the same decision. Thus, a user, searching for information, has to guess the classifier's choice to decide where to look for, and will typically have to look in a number of places.

[5]Look at http://docs.yahoo.com/info/suggest/appropriate.html to see how Yahoo! implements this rule.

- each non-root node in the hierarchy has only one parent node. This enforces a tree structure on the hierarchy.

Example 2 (Amazon book directory). Many search engines like Google, Yahoo as well as many eCommerce vendors, like Amazon, offer mechanisms to search for relevant items. This is the case, for instance, of the web directory catalogue for books (among other items) used in Amazon. At present Amazon has 35 main subjects. Books are inserted by the classifier in the web directory, and users browse such classification hierarchy to access the books they are interested in.

In Amazon, as in DDC, books can be classified both in leaf and non-leaf nodes[6], following the "Get Specific" criterion, but also the "Related Directory" criterion[7], when the classifier browses through the hierarchy looking for an appropriate category that lists similar documents. In this classification hierarchy, a book can be often reached from different paths of the hierarchy, thus providing efficient tools to arrive at items of interest using different perspectives.

In the following we present an example of classification for a software programming book in the Amazon Book Web Directory. The book title is "Enterprise Java Beans, Fourth Edition". In the current Amazon book directory[8], the example title can be found through two different search paths (see Figure 2), namely:

`Subjects` → `Business and Investing` → `Small Business and Entrepreneurship` → `New Business Enterprises`

`Subjects` → `Computers and Internet` → `Programming` → `Java Language` → `Java Beans`

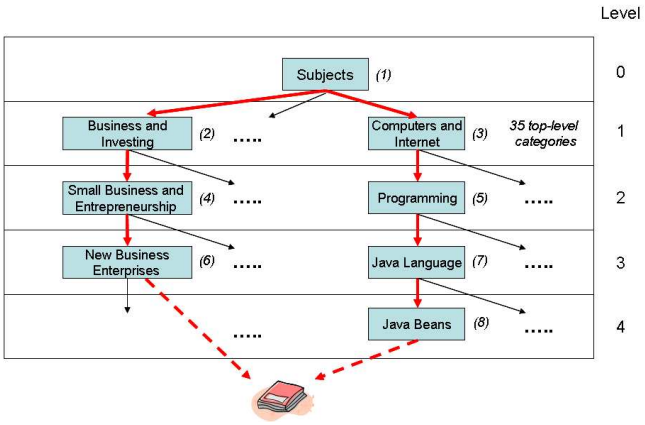


Figure 2: Amazon Book Directory

From the brief presentation and from the two specific examples we can see that Web catalogues are more open than classifications like Dewey. In fact, their aim is not to try to position a resource in a unique position, but rather to position it in such a way, that a user, who navigates the catalogue, will be facilitated to find appropriate or similar resources related to a given topic.

Formal Classifications

Let us use the two examples above to present and discuss a number of characteristics that are relevant to classifications and that need to be considered in a formal theory of classification.

Let us start from the characteristics of edges. People consider classifications top down. Namely, when classifying or searching for a document first upper level nodes are considered, and then, if these nodes are too general for the given criteria, lower level nodes may also be inspected. Child nodes in a classification are always considered in the context of their parent nodes, and therefore *specialize* the meaning of the parent nodes. In a classification there are two possible meaningful interrelationships between parent and child nodes as shown on Figure 3:

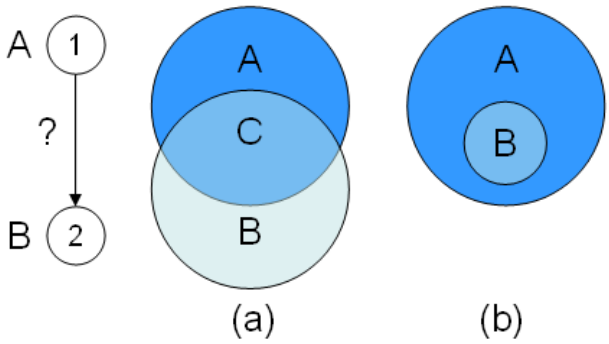


Figure 3: Edge semantics for formal classifications

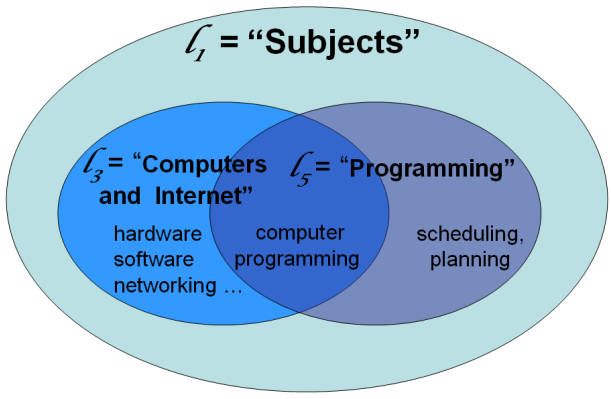


Figure 4: Example of general intersection

- Case (a) represents edges expressing the "general intersection" relation, and, intuitively, the meaning of node 2 is area C, which is the intersection of areas A and B.

[6]Amazon implements it by assigning to non-leaf nodes a leaf node labeled "General", where items related to the non-leaf nodes are classified

[7]Look at http://www.google.com/dirhelp.html#related to see how Google implements this rule

[8]See http://www.amazon.com, April 2005.

For instance, in our Amazon example, the edge in Figure 2 `Computers and Internet` → `Programming` codifies all the items that are in common (see Figure 4) to the categories `Computers and Internet` (i.e., hardware, software, networking, etc) and `Programming` (i.e., scheduling, planning, computer programming, web programming, etc). This kind of edges are also present in library systems, such as DDC, at lower levels of the hierarchy where different facets of a particular parent category are considered.

- Case (b) represents a more specific case where the child node is "subsumed by" the parent node. In this case the meaning of node 2 is area B. This kind of edges is also called an "is-a" edge. Note that in this case, differently from case (a), node A does not influence what is classified in node B.

  Many edges in DDC impose the "is-a" relation, in particular in the higher levels of the hierarchy. Also some edges in the Amazon book directory impose the "is-a" links, the most obvious ones are the edges from the root category.

Notice that, in the case of edges leading to the same resource the "general intersection" relation must hold for *all* the categories in all the different paths. The latter fact can be used to improve the classification representation: either by trying to prohibit this situation (if the goal is to classify unambiguously a resource, as it happens in a library classification, such as DDC) or by enhancing this kind of situation (if the goal is improving the recall of relevant resources, as it happens in a web catalogue, such as Amazon).

Let us now move to consider the characteristics of labels. As from Definition 1, the concept of a specific node is described by a label expressed in words and, possibly, separators between them. The node labels possess interesting structure, relevant to formal classification hierarchies:

- Natural language labels are composed by atomic elements, namely words. These words can be analyzed in order to find all their possible basic forms and eventual multiple senses, i.e., the way in which the word can be interpreted. In this paper, we use WordNet (Miller 1998) to retrieve word senses[9], however, in practice, a different thesaurus can be used. For example the word "Java" in the label "Java Language" in Figure 2 possesses different equivalent forms (e.g., Java, java) and three different senses:
  1. an island in Indonesia;
  2. a beverage consisting of an infusion of ground coffee beans; and
  3. an object oriented programming language.
- Words are combined to build complex concepts out of the atomic elements. Consider for example the labels `Computers and Internet` and `Java Language` in Figure 2. The combination of natural language atomic elements is used by classifier to aggregate (like in `Computers and Internet`) or disambiguate atomic concepts (like in `Java Language`, where the sense of the word `Java` that denotes "an island in Indonesia" together with the sense "a type of coffee" can be discarded while the correct sense of "an object oriented programming language" is maintained).
- Natural language labels make use of the structure of the classification hierarchy to improve the semantic interpretation associated to a given node. We call this property *parental contextuality* of a node. For instance the sense of words composing labels of different nodes in an hierarchy path can be incompatible; thus the correct meaning of a particular word in a specific label can be disambiguated by considering the senses of the words in some labels along the path. For example, in the path `Java Languages` → `Java Bean`, the possible correct (but wrong) sense of `Java Bean` as "a particular type of coffee bean" can be pruned by the classifier taking into account the meaning of the parent node's label, `Java Languages`.

Let us see how we can convert classifications into a new structure, which we call a *Formal Classification* (FC), more amenable to automated processing:

Definition 2 (Formal Classification) *A Formal Classification is a rooted tree described by a triple $H^F = \langle C, E, l^F \rangle$ where C is a finite set of nodes, E is a set of edges on C, and l^F is a function from C to a set L^F of labels expressed in a Propositional Description Logic language L^C.*

As it can be noticed, the key step is that in FCs labels are substituted by labels written in a formal logical language. In the following we will call L^C, the *Concept Language*. We use a Propositional Description Logic language for several reasons. First, we move from an ambiguous language to a formal language with clear semantics. Second, given its set-theoretic interpretation, L^C "maps" naturally to the real world semantics. For instance, the atomic proposition p =`computer` denotes "the set of machines capable of performing calculations automatically". Third, natural language labels are usually short expressions or phrases having simple syntactical structure. Thus no sophisticated natural language processing and knowledge representation techniques are required – a phrase can be often converted into a formula in L^C with no or little loss in the meaning. Forth, a formula in L^C can be converted into an equivalent formula in a propositional logic language with boolean semantics. Thus a problem expressed in L^C can therefore be converted into a *propositional satisfiability problem*[10].

Apart from the atomic propositions, the language L^C includes logical operators, such as *conjunction* (denoted by $\sqcap$), *disjunction* (denoted by $\sqcup$), and *negation* ($\neg$); as well as comparison operators: *more general* ($\sqsupseteq$), *more specific* ($\sqsubseteq$), and *equivalence* ($\equiv$). In the following we will also say that A *subsumes* B, if $A \sqsupseteq B$; and we will also say that A is *subsumed* by B, if $A \sqsubseteq B$. The interpretation of the operators is the standard set-theoretic interpretation.

[9] We may change the actual senses of a word from WordNet for the sake of presentation.

[10] For translation rules from a Propositional Description Logic to a Propositional Logic, see (Bouquet, Serafini, & Zanobini 2003; Giunchiglia, Shvaiko, & Yatskevich 2004).

We build FCs out of classifications by translating, using natural language processing techniques, natural language labels, l_i's, into concept language labels, l_i^F's. For lack of space we do not describe here how we perform this step. The interested reader is referred to (Magnini, Serafini, & Speranza 2003). As an example, recall the classification example shown on Figure 2. For instance, the label `Java beans` of node n_8 is translated into the following expression:

$$l_8^F = (\texttt{Java}^1 \sqcup \texttt{Java}^2 \sqcup \texttt{Java}^3) \sqcap (\texttt{Bean}^1 \sqcup \texttt{Bean}^2) \quad (1)$$

where $\texttt{Java}^1$ denotes the Java island, $\texttt{Java}^2$ is a brewed coffee, $\texttt{Java}^3$ is the object oriented programming language Java, $\texttt{Bean}^1$ is a kind of seeds, and $\texttt{Bean}^2$ is a Java technology related term. The disjunction $\sqcup$ is used to codify the fact that `Java` and `Bean` may mean different things. The conjunction $\sqcap$ is used to codify that the meaning of `Java beans` must take into account what `Java` means *and* what `Beans` mean.

As it is mentioned above, some senses of a word in a label may be incompatible with the senses of the other words in the label, and, therefore, these senses can be discarded. A way to check this in L^C is to convert a label into *Disjunctive Normal Form* (DNF). A formula in DNF is a disjunction of conjunctions of atomic formulas or negation of atomic formulas, where each block of conjunctions is called a *clause* (Mendelson 4th ed London 1997). Below is the result of conversion of Formula 1 into DNF:

$$\begin{aligned} l_8^F = \quad & (\texttt{Bean}^1 \sqcap \texttt{Java}^1) \sqcup (\texttt{Bean}^1 \sqcap \texttt{Java}^2) \sqcup \\ & (\texttt{Bean}^1 \sqcap \texttt{Java}^3) \sqcup (\texttt{Bean}^2 \sqcap \texttt{Java}^1) \sqcup \\ & (\texttt{Bean}^2 \sqcap \texttt{Java}^2) \sqcup (\texttt{Bean}^2 \sqcap \texttt{Java}^3) \end{aligned} \quad (2)$$

The first clause in Formula 2 (i.e., $(\texttt{Bean}^1 \sqcap \texttt{Java}^1)$) can be discarded, as there is nothing in common between seeds and the island. The second clause, instead, is meaningful – it denotes the coffee seeds. Analogously, clauses 3, 4 and 5 are discarded and clause 6 is preserved. The final formula for the label of node n_8 therefore becomes:

$$l_8^F = (\texttt{Bean}^1 \sqcap \texttt{Java}^2) \sqcup (\texttt{Bean}^2 \sqcap \texttt{Java}^3) \quad (3)$$

Note, that sense $\texttt{Java}^1$ is pruned away in the final formula as it has nothing to do with any sense of the word "bean". Analogously, all the other labels in the classification shown on Figure 2 are translated into expressions in L^C and further simplified. At this point, the "converted" Classification represents a FC.

Note, that each clause in DNF represents a distinct meaning encoded into the label. This fact allows both agents and classifiers to operate on meanings of labels, and not on meanings of single words.

Normalized Formal Classifications

As discussed earlier, in classifications, child nodes are considered in the context of their parent nodes. We formalize this notion of parental context in a FC following the definition of concept at a node from (Giunchiglia, Shvaiko, & Yatskevich 2004):

Definition 3 (Concept at a node) *Let H^F be a FC and n_i be a node of H^F. Then, the concept at node n_i, written C_i, is its label l_i^F if n_i is the root of H^F, and, otherwise, it is the conjunction of the label of n_i and the concept at node n_j, which is the parent of n_i. In formulas:*

$$C_i = \begin{cases} l_i^F & \text{if } n_i \text{ is the root of } H^F \\ l_i^F \sqcap C_j & \text{if } n_i \text{ is a non-root node of } H^F, \\ & \text{where } n_j \text{ is the parent of } n_i \end{cases}$$

Applying Definition 3 recursively, we can compute the concept at any non-root node n_i as the conjunction of the labels of all the nodes on the path from the root of H^F to n_i:

$$C_i = l_1^F \sqcap l_2^F \sqcap \ldots \sqcap l_i^F \quad (4)$$

The notion of concept at a node explicitly captures the classification semantics. Namely, the interpretation of the concept at a node is the set of objects that the node and all its ascendants have in common (see Figure 3). From the classification point of view, the concept at a node defines what (class of) documents can be classified in this node.

The definition of concept at a node possesses a number of important properties relevant to classification:

Property C.1: each C_i codifies both the label of n_i and the path from the root to n_i. There are two important consequences of this: first, it allows it to prune away irrelevant senses along the path; and, if converted to DNF, C_i represents the union of all the possible distinct meanings of a node in the FC's tree.

Recall the Amazon running example. According to Formula 4, the concept at node n_8 is:

$C_8 = (\texttt{Subject}^*) \sqcap (\texttt{Computer}^* \sqcup \texttt{Internet}^*) \sqcap (\texttt{Programming}^*) \sqcap (\texttt{Java}^* \sqcap \texttt{Language}^*) \sqcap (\texttt{Java}^* \sqcap \texttt{Bean}^*)$ [11]

The possible correct (but wrong) sense $(\texttt{Bean}^1 \sqcap \texttt{Java}^2)$ as "a particular type of coffee bean" (the first clause in Formula 3) can be pruned by converting the concept at node n_8 into DNF, which contains the clause $(\texttt{Language}^1 \sqcap \texttt{Java}^2 \sqcap \texttt{Bean}^1)$ and checking it as a propositional satisfiability problem: since the meaning of $\texttt{Language}^1$ is "incompatible" with $\texttt{Java}^2$ the expression results into an inconsistency.

Property C.2: each C_i has a normal form. In fact it is always possible to transform each C_i in Conjunctive Normal Form (CNF) namely a conjunction of disjunctions of atomic formulas or negation of atomic formulas (Mendelson 4th ed London 1997). Therefore C_i codifies in one logical expression *all* the possible ways of conveying the same concept associated to a node.

We use the notion of the concept of a node to define a further new structure which we call *Normalized Formal Classification* (NFC):

Definition 4 (Normalized Formal Classification) *A Normalized Formal Classification is a rooted tree described by a triple $H^N = \langle C, E, l^N \rangle$ where C is a finite set of nodes, E is a set of edges on C, and l^N is a function from C to a set L^N of concepts at nodes.*

[11] We write $\texttt{X}^*$ to denote the disjunction of all the senses of X.

Also the proposed NFC possesses a number of important properties relevant to classification:

Property NFC.1: when all C_i are expressed in CNF (see property C.2), all the nodes expressing *semantically equivalent* concepts will collapse to the same CNF expression. Even when two computed concepts are not equivalent, the comparison of the two CNF expressions will provide enhanced similarity analysis capability to support both classification and query-answering tasks.
Following our example, the normalized form of the concept at node n_8 with the path (in natural language):

`Subjects` → `Computers and Internet` → `Programming` → `Java Language` → `Java Beans`

will be equivalent, for instance, to the concept associated to a path like:

`Topic` → `Computer` → `Internet` → `Programming` → `Languages` → `Java` → `Java Beans`

and similar (i.e., be more general, or more specific) to (say):

`Discipline` → `Computer Science` → `Programming languages` → `Java` → `J2EE` → `Java Beans`

Property NFC.2: any NFC is a taxonomy, in the sense that for any non-root node n_i and its concept C_i, the concept C_i is always subsumed by C_j, where n_j is the parent node of n_i. We claim that NFCs are the "correct" translations of classifications into ontological taxonomies as they codify the intended semantics/use of classifications. Notice that, under this assumption, in order to capture the classification semantics no expressive ontological languages are needed, and a Propositional Description Logic is sufficient. In this respect our work differs substantially from the work described in (Magnini, Serafini, & Speranza 2003).
Consider in our running Amazon example the path in the natural language classification:

`Subject` → `Computers and Internet` → `Programming`

As described in section "Classifications", this path contains a link expressing the "general intersection" relation, namely the link is `Computers and Internet` → `Programming` (see Figure 4). The same relation is maintained when we move to FCs. In our notation: $l_1^F = \texttt{Subject}^*$, $l_3^F = (\texttt{Computer}^* \sqcup \texttt{Internet}^*)$, $l_5^F = \texttt{Programming}^*$. But, when we move to the NFC for the given example, our elements become: $C_1 = l_1^F$; $C_3 = l_1^F \sqcap l_3^F$; $C_5 = l_1^F \sqcap l_3^F \sqcap l_5^F$; and the only relation holding between successive element is the subsumption.

The above properties of both C_i and NFC have interesting implications in classification and query answering, as described in the next Section.

Document classification and query answering

We assume that each document d is assigned an expression in L^C, which we call the *document concept*, written C^d. The assignment of concepts to documents is done in two steps: first, a set of document's keywords is retrieved using text mining techniques (see, for example, (Sebastiani 2002)); the keywords are then converted into a corresponding concept using the same techniques used to translate natural language labels into concept language labels.

There exists a number of approaches to how to classify a document. In one such approach a document is classified only in one node (as in DDC), in another approach it may be classified under several nodes (as in Amazon). However, in most cases, the general rule is to classify a document in the node or in the nodes that most specifically describe the document, i.e., to follow the "Get Specific" criterion discussed in section "Classifications". In our approach, we allow for a document to be classified in more than one node, and we also follow the "Get Specific" criterion. We express these criteria, in a formal way, as follows:

Definition 5 (Classification Set) *Let H^N be a NFC, d be a document, and C^d be the concept of d. Then, the classification set for d in H^N, written Cl^d, is a set of nodes $\{n_i\}$, such that for any node $n_i \in Cl^d$ the following two conditions hold:*

1. *the concept at node n_i is more general than C^d, i.e. $C^d \sqsubseteq C_i$; and*
2. *there is no such node n_j ($j \neq i$), whose concept at node is more specific than C_i and more general than C^d.*

Document d is classified in all the nodes from the set Cl^d in Definition 5.

Suppose we are given two documents: a book on Java programming (d_1) and an article on high tech entrepreneurship (d_2). Suppose now that these documents are assigned the following concepts: $C_1^d = \texttt{Java}^3 \sqcap \texttt{Programming}^2$, and $C_2^d = \texttt{High\_tech}^1 \sqcap \texttt{Venture}^3$, where $\texttt{Java}^3$ is the programming language, $\texttt{Programming}^2$ is computer programming, $\texttt{High\_tech}^1$ is "highly advanced technological development", and $\texttt{Venture}^3$ is "a commercial undertaking that risks a loss but promises a profit". Intuitively, C_1^d is more specific than the concept at the node labeled `Java language` in the classification shown on Figure 2. In fact, logical inference confirms the intuition, namely it is possible to show that the following relation holds: $C_1^d \sqsubseteq C_7$. It is also possible to show that the second condition of Definition 5 holds for node n_7. Thus, document d_1 is classified in node n_7. Analogously, it can be shown that the classification set for d_2 is composed of the single node n_6. For lack of space we do not show the full formulas and the proofs of these statements.

Moving to query answering, when a user searches for a document, she defines a set of keywords or a phrase, which is then converted into an expression in L^C using the same techniques discussed in section "Formal Classifications". We call this expression, a *query concept*, written C^q. We

define the answer A^q to a query q as the set of documents, whose concepts are more specific than the query concept C^q:

$$A^q = \{d | C^d \sqsubseteq C^q\} \quad (5)$$

Searching directly on all the documents may become prohibitory expensive as classifications may contain thousands and millions of documents. NFCs allow us to identify the maximal set of nodes which contain *only* answers to a query, which we call, the *sound classification answer* to a query (written N_s^q). We compute N_s^q as follows:

$$N_s^q = \{n_i | C_i \sqsubseteq C^q\} \quad (6)$$

In fact, as $C^d \sqsubseteq C_i$ for any document d classified in any node $n_i \in N_s^q$, and $C_i \sqsubseteq C^q$, then $C^d \sqsubseteq C^q$. Thus, all the documents classified in the set of nodes N_s^q belong to the answer A^q (see Formula 5).

We extend N_s^q by adding nodes, which constitute the classification set of a document d, whose concept is $C^d = C^q$. We call this set, the *query classification set*, written Cl^q; and we compute it following Definition 5. In fact, nodes in Cl^q may contain documents satisfying Formula 5, for instance, documents whose concepts are equivalent to C^q.

Suppose, for instance, that a user defines the following query to the Amazon NFC: $C^q = \texttt{Java}^3 \sqcup \texttt{COBOL}^1$, where $\texttt{COBOL}^1$ is "common business-oriented language". It can be shown, that $N_s^q = \{n_7, n_8\}$ (see Figure 2 for the Amazon classification). However, this set does not include node n_5, which contains the book "Java for COBOL Programmers (2nd Edition)". Node n_5 can be identified by computing the query classification set for query q, which in fact consists of the single node n_5, i.e. $Cl^q = \{n_5\}$. However, n_5 may also contain irrelevant documents.

Thus, for any query q, a user can compute a sound query answer A_s^q by taking the union of two sets of documents: the set of documents which are classified in the set of nodes N_s^q (computed as $\{d \in n_i | n_i \in N_s^q\}$); and the set of documents which are classified in the nodes from the set Cl^q and which satisfy Formula 5 (computed as $\{d \in n_i | n_i \in Cl^q, C^d \sqsubseteq C^q\}$). We have therefore:

$$A_s^q = \{d \in n_i | n_i \in N_s^q\} \cup \{d \in n_i | n_i \in Cl^q, C^d \sqsubseteq C^q\} \quad (7)$$

Under the given definition, the answer to a query is not restricted to the documents classified in the nodes, whose concepts are the "closest" match to the query. Documents from nodes, whose concepts are more specific than the query are also returned. For instance, a result for the above mentioned query may also contain documents about Java beans.

Note, that the structure of a NFC (i.e., the edges) is *not* considered neither during document classification nor during query answering. In fact, given the proposed classification algorithm, the edges information becomes redundant, as it is implicitly encoded in the concepts at the nodes. We say *implicitly* because there may be more than one way to "reconstruct" a NFC resulting into the same set of concepts at nodes. But, all the possible NFCs are equivalent, in the sense that the same set of documents is classified into exactly the same set of nodes.

The algorithms presented in this section are sound and complete in the document classification part, as Propositional Logic allows for sound and complete reasoning on documents according to Definition 5. The proposed solution for query answering is sound but not complete as $A_s^q \subseteq A^q$. For lack of space we do not provide evidence of the incompleteness property of the solution.

Related Work

In our work we adopt the notion of the concept at node as first introduced in (Giunchiglia & Shvaiko 2003) and further elaborated in (Giunchiglia, Shvaiko, & Yatskevich 2004). Moreover, the notion of label of a node in a FC, semantically corresponds to the notion of the concept of a label introduced in (Giunchiglia, Shvaiko, & Yatskevich 2004). In (Giunchiglia, Shvaiko, & Yatskevich 2004) these notions play the key role in the identification of semantic mappings between nodes of two schemas. In this paper, these are the key notions needed to define NFCs.

This work as well as the work in (Giunchiglia & Shvaiko 2003; Giunchiglia, Shvaiko, & Yatskevich 2004) mentioned above is crucially related and depends on the work described in (Bouquet, Serafini, & Zanobini 2003; Magnini, Serafini, & Speranza 2003). In particular, in (Bouquet, Serafini, & Zanobini 2003), the authors, for the first time ever, introduce the idea that in classifications, natural language labels should be translated in logical formulas, while, in (Magnini, Serafini, & Speranza 2003), the authors provide a detailed account of how to perform this translation process. The work in (Giunchiglia & Shvaiko 2003; Giunchiglia, Shvaiko, & Yatskevich 2004) improves on the work in (Bouquet, Serafini, & Zanobini 2003; Magnini, Serafini, & Speranza 2003) by understanding the crucial role that concepts at nodes have in matching heterogeneous classifications and how this leads to a completely new way to do matching. As a matter of fact the work in (Giunchiglia & Shvaiko 2003) classifies the work in (Bouquet, Serafini, & Zanobini 2003; Giunchiglia & Shvaiko 2003; Giunchiglia, Shvaiko, & Yatskevich 2004; Magnini, Serafini, & Speranza 2003) as *semantic matching* and distinguishes it from all the previous work, classified under the heading *syntactic matching*. This paper, for the first time, recognizes the crucial role that the ideas introduced in (Bouquet, Serafini, & Zanobini 2003; Giunchiglia & Shvaiko 2003; Giunchiglia, Shvaiko, & Yatskevich 2004; Magnini, Serafini, & Speranza 2003) have in the construction of a new theory of classification, and in introducing the key notion of FC.

A lot of work in information theory, and more precisely on formal concept analysis (see for instance (Wille 1992)) has concentrated on the study of concept hierarchies. NFCs are what in formal concept analysis are called concept hierarchies with no attributes. The work in this paper can be considered as a first step towards providing a computational theory of how to transform the "usual" natural language classifications into concept hierarchies. Remember that concept hierarchies are ontologies which are trees where parent nodes subsume their child nodes.

The classification and query answering algorithms, proposed in this paper, are similar to what in the Description Logic (DL) community is called *realization* and *retrieval* respectively. The fundamental difference between the two approaches is in that in DL the underlying structure for classification is not predefined by the user, but is build bottom-up from atomic concepts by computing the subsumption partial ordering. Interested readers are referenced to (Horrocks *et al*. 2004), where the authors propose sound and complete algorithms for realization and retrieval.

In Computer Science, the term *classification* is primarily seen as the *process* of arranging a set of objects (e.g., documents) into *categories* or *classes*. There exist a number of different approaches which try to build classifications *bottom-up*, by analyzing the contents of documents. These approaches can be grouped in two main categories: *supervised classification*, and *unsupervised classification*. In the former case, a small set of training examples needs to be prepopulated into the categories in order to allow the system to automatically classify a larger set of objects (see, for example, (G.Adami, P.Avesani, & D.Sona 2003; Nigam *et al*. 2000)). The latter approach uses various machine learning techniques to classify objects, for instance, data clustering (Jain, Murty, & Flynn 1999). There exist some approaches that apply (mostly) supervised classification techniques to the problem of documents classification into hierarchies (Koller & Sahami 1997; Sun & Lim 2001). The classifications built following our approach are better and more natural than those built following these approaches. They are in fact constructed *top-down*, as chosen by the user and not constructed bottom-up, as they come out of the document analysis. Notice how in this latter case the user has no or little control over the language used in classifications. Our approach has the potential, in principle, to allow for the automatic classification of the (say) Yahoo! documents into the Yahoo! directories. Some of our current work is aimed at testing the feasibility of our approach with very large sets of documents.

Conclusions

In this paper we have introduced the notion of Formal Classification, namely of a classification where labels are written in a propositional concept language. Formal Classifications have many advantages over standard classifications all deriving from the fact that formal language formulas can be reasoned about far more easily than natural language sentences. In this paper we have highlighted how this can be done to perform query answering and document classification. However much more can be done. Our future work includes the development of a sound and complete query answering algorithm; as well as the development and evaluation of tools that implement the theoretical framework presented in this paper. There are two tools of particular importance, namely the document classifier and query answering tools, which will provide the functionality described in this paper. The performance of the tools will then be compared to the performance of the most advanced heuristics based approaches. Yet another line of research will be the development of a theoretical framework and algorithms allowing for the interoperability between NFCs. The latter particularly includes distributed query answering and multiple document classification under sound and complete semantics.

References

Baader, F.; Calvanese, D.; McGuinness, D.; Nardi, D.; and Patel-Schneider, P. 2003. *The Description Logic Handbook : Theory, Implementation and Applications*. Cambridge University Press.

Bouquet, P.; Serafini, L.; and Zanobini, S. 2003. Semantic coordination: a new approach and an application. *In Proc. of the 2nd International Semantic Web Conference (ISWO'03). Sanibel Islands, Florida, USA.*

G.Adami; P.Avesani; and D.Sona. 2003. Clustering documents in a web directory. *In Proceedings of Workshop on Internet Data management (WIDM-03)*.

Giunchiglia, F., and Shvaiko, P. 2003. Semantic matching. *"Ontologies and Distributed Systems" workshop, IJCAI*.

Giunchiglia, F.; Shvaiko, P.; and Yatskevich, M. 2004. S-match: An algorithm and an implementation of semantic matching. *In Proceedings of ESWS'04*.

Gordon, A. Second edition, 1999. *Classification*. Monographs on Statistics and Applied Probability. Chapman-Hall/CRC.

Horrocks, I.; Li, L.; Turi, D.; and Bechhofer, S. 2004. The instance store: DL reasoning with large numbers of individuals. In *Proc. of the 2004 Description Logic Workshop (DL 2004)*, 31–40.

Jain, A. K.; Murty, M. N.; and Flynn, P. J. 1999. Data clustering: a review. *ACM Computing Surveys* 31(3):264–323.

Koller, D., and Sahami, M. 1997. Hierarchically classifying documents using very few words. In Fisher, D. H., ed., *Proceedings of ICML-97, 14th International Conference on Machine Learning*, 170–178. Nashville, US: Morgan Kaufmann Publishers, San Francisco, US.

Magnini, B.; Serafini, L.; and Speranza, M. 2003. Making explicit the semantics hidden in schema models. *In: Proceedings of the Workshop on Human Language Technology for the Semantic Web and Web Services, held at ISWC-2003, Sanibel Island, Florida*.

Mendelson, A. 4th ed. London, 1997. *Introduction to Mathematical Logic*. Chapman-Hall.

Miller, G. 1998. *WordNet: An electronic Lexical Database*. MIT Press.

Nigam, K.; McCallum, A. K.; Thrun, S.; and Mitchell, T. M. 2000. Text classification from labeled and unlabeled documents using EM. *Machine Learning* 39(2/3):103–134.

Sebastiani, F. 2002. Machine learning in automated text categorization. *ACM Computing Surveys* 34(1):1–47.

Sun, A., and Lim, E.-P. 2001. Hierarchical text classification and evaluation. In *ICDM*, 521–528.

Wille, R. 1992. Concept lattices and conceptual knowledge systems. *Computers and Mathematics with Applications* 23:493–515.

First-Orderized ResearchCyc : Expressivity and Efficiency in a Common-Sense Ontology

Deepak Ramachandran[1] **Pace Reagan**[2] **and Keith Goolsbey**[2]
[1]Computer Science Department, University of Illinois at Urbana-Champaign, Urbana, IL 61801
dramacha@uiuc.edu
[2]Cycorp Inc., 3721 Executive Center Drive, Suite 100, Austin, TX 78731
{pace,goolsbey}@cyc.com

Abstract

Cyc is the largest existing common-sense knowledge base. Its ontology makes heavy use of higher-order logic constructs such as a context system, first class predicates, etc. Many of these higher-order constructs are believed to be key to Cyc's ability to represent common-sense knowledge and reason with it efficiently. In this paper, we present a translation of a large part (around 90%) of the Cyc ontology into First-Order Logic. We discuss our methodology, and the tradeoffs between expressivity and efficiency in representation and reasoning. We also present the results of experiments using VAMPIRE, SPASS, and the E Theorem Prover on the first-orderized Cyc KB. Our results indicate that, while the use of higher-order logic is not essential to the representability of common-sense knowledge, it greatly improves the efficiency of reasoning.

1 Introduction

ResearchCyc™ is a version of Cycorp Inc.'s Cyc®[1] Knowledge Base - the world's largest general common-sense ontology and reasoning engine. ResearchCyc is available under a free license, and consists of 1,074,484 assertions in a language of 122,658 symbols (not including strings or numbers). By contrast, the IEEE Suggested Upper Merged Ontology (Niles & Pease 2001) consists of 60,000 axioms over 20,000 terms.

A significant feature of ResearchCyc's design is the incorporation of higher-order assertions in its KB. (For the rest of this paper we will use the term "higher-order" to mean any feature beyond ordinary first-order logic, like a context system or quantification over predicates.) The reasons for this are both philosophical and pragmatic; it is widely believed that a complete specification of what is understood to be "common-sense knowledge" requires some kind of higher-order features. For example, Boolos (Boolos 1984) discusses two sentences that cannot be represented in a logic without predicate quantification (*non-firstorderizable*):

1. Some critics admire only one another.
2. Some of Fianchetto's men went into the warehouse unaccompanied by anyone else.

[1]Cyc is a registered trademark and ResearchCyc and OpenCyc are trademarks of Cycorp, Inc.

The second sentence is non-firstorderizable if we consider "anyone else" to mean anyone who was not in the group of Fianchetto's men who went into the warehouse.

Another reason for the use of these higher-order constructs is the ability to represent knowledge succinctly and in a form that can be taken advantage of by specialized reasoning methods. The validation of this claim is one of the contributions of this paper.

In this paper, we present a tool for translating a significant percentage (around 90%) of the ResearchCyc KB into First-Order Logic (a process we call *FOLification*). Our methods involve a number of non-trivial transformations of the higher-order constructs appearing in ResearchCyc into FOL. In some cases (e.g. axiom schemata), the assertions could be translated precisely without any loss of semantics, but with an increase in the size of the representation (both the number of symbols and axioms). In other cases (e.g. contexts) only an approximate translation could be made. We describe our techniques in detail and the tradeoffs involved in expressivity and efficiency. Our final FOLified ResearchCyc KB consisted of 1,253,117 axioms over 132,116 symbols (not including strings or numbers).

We also present the preliminary results of some experiments performed on the FOLified KB using VAMPIRE (Riazanov 2003), SPASS (Weidenbach 1999), and the E theorem prover (Schulz 2002) with a sample collection of common-sense queries typical for ResearchCyc. The ATP systems performed orders of magnitude more slowly than the ResearchCyc inference engine that is custom built for Cyc's ontology and thus able to reason with its higher-order representations directly. From these results, we conclude that the types of problems Cyc is designed to handle differ substantially from the types of problems that the ATP systems are designed to handle. To our knowledge, this is the largest ever set of axioms upon which first-order theorem proving has been attempted. Our results are open to interpretation and we discuss them in the conclusion.

The rest of this paper is organized as follows: In Section 2 we provide some background. In Section 3 we discuss our FOLification procedure. In Section 4 we discuss the design of the CYC inference engine. Section 5 presents our preliminary experimental results. Finally in Section 6 we discuss conclusions and future work.

2 Background

We assume that the reader is familiar with all the standard terminology from the fields of Knowledge Representation and Automated Reasoning.

The Cyc common-sense knowledge base has been in development since 1984, first as a project at MCC, and since 1994 by Cycorp Inc. (*www.cyc.com*). A description of the motivation and goals of the Cyc project is given in (Lenat & Guha 1990). The complete Cyc knowledge base is under a proprietary license, but recently Cycorp has released a significant proportion of the KB to the academic community through the ResearchCyc project (*research.cyc.com*). ResearchCyc is available under a restricted but free license to all research institutions and universities. The eventual goal is to make almost all of the Cyc KB, except for certain sensitive sections, available in ResearchCyc. OpenCyc™(*www.opencyc.org*) is the open-source version of a subset of the Cyc KB (mostly the upper ontology) freely distributed under the Creative Commons license. In the rest of this paper, for brevity we will refer to all three KBs simply as Cyc, except where we wish to distinguish among them.

CycL is the declarative knowledge representation language used to store Cyc's KB. It has a Lisp-like syntax, with each atom and each term appearing within a pair of parentheses. As mentioned before, it is higher-order and has all the standard logical connectives and operators. The Cyc assertions and queries we use in the rest of this paper are written in simplified CycL.

TPTP (Thousands of Problems for Theorem Provers) (Sutcliffe & Suttner 1998) is the most popular library of test problems for automated theorem proving (ATP) systems and is the basis for the CADE ATP System Competition (Pelletier, Sutcliffe, & Suttner 2002). Our FOLification tool translates CycL into the TPTP format. We chose this format because many theorem provers accept it as input, and there exist tools for converting TPTP files into virtually every other first-order format.

3 First-Orderization of Higher-Order Constructs

In this section, we describe some of the transformations used to convert Cyc's higher-order KB into FOL.

It should be noted that our use of the term "higher-order" is rather loose; it signifies any conceivable extension to the usual syntactic restrictions of FOL: variable-arity predicates, sentences as arguments, etc.

3.1 Contexts

Contexts have been viewed both as a scheme for acheiving generality (McCarthy 1986) and locality (Giunchiglia 1993). ResearchCyc's context system enables both.

Each assertion in the KB is associated with a *microtheory* ("mt") that determines its context. The microtheories are arranged in a hierarchy, with each microtheory having access to all the assertions in its parents.

Microtheories are implemented by extending the language with the modal operator `ist`(k, φ)("is true") to express that the assertion φ is true in the context k. For example, consider the context of world mythology, in which we assert the fact that vampires fear garlic:

```
ist(WorldMythologyMt, (Vampire(X) ∧
Garlic(Y)) ⇒ (feelsTowardsObject ?X ?Y
Fear positiveAmountOf))
```

CycL has a predicate `genlMt` to assert that a microtheory is a child of another in the hierarchy (and thus inherits all its assertions):

```
genlMt(WorldMythologyMt, HumanActivi-
tiesMt)
```

The closest analogue to Cyc's microtheory system in the literature is Buvac's quantificational theory of context (Buvac 1996).

There are two methods for handling microtheories in pure first-order logic. One is to treat each context as a separate module, choose at inference time which modules are required, and compile them into a single flat KB. This is the approach used by the Suggested Upper Merged Ontology (Niles & Pease 2001).

A more sophisticated scheme is to approximate the `ist` modality with an `mtVisible` predicate. For example,

```
mtVisible(WorldMythologyMt) ⇒ ((Vam-
pire(X) ∧ Garlic(Y)) ⇒ (feelsTowardsOb-
ject ?X ?Y Fear positiveAmountOf))
```

Contexts can then inherit from each other:

```
mtVisible(WorldMythologyMt) ⇒ mtVisi-
ble(HumanActivitiesMt)
```

Criteria can be imposed for selecting certain contexts:

$\forall k[(\texttt{mtVisible}(k) \Rightarrow \exists X \texttt{Dragon}(X))$
$\Rightarrow \texttt{FictionalContext}(k)]$

Both techniques mentioned above have one limitation: the inability to represent inter-contextual reasoning. In the module-based method there are no means of referring to other contexts. In the `mtVisible` method, while it is possible for assertions to refer to other contexts, the meaning is usually not what is intended. For example, suppose context k_1 knows that axiom ϕ is true in context k_2. If we try to translate this as:

`mtVisible(`k_1`) ⇒ (mtVisible(`k_2`) ⇒`ϕ`)`

This is equivalent to the statement:

`mtVisible(`k_2`) ⇒ (mtVisible(`k_1`) ⇒`ϕ`)`

which was not intended at all.

Our translation tool uses a combination of both methods. In general, each assertion is prefixed with the `mtVisible` condition for the microtheory it belongs to. But in the case of

some microtheories being universally or almost universally visible, we use the modular approach. For example, when the assertion is from either `UniversalVocabularyMt` or `BaseKB`, then it is FOLified without adding the `mtVisible` predicate at all, since assertions from these microtheories are visible to practically every query.

3.2 Ontology

The backbone of Cyc's ontology of types and objects is the *genls* hierarchy. It is analogous to the `genlMt` hierarchy of microtheories. Every `Thing` in Cyc is either an `Individual` or a `Collection`. Both `Individual`s and `Collection`s can be instances of `Collection`s. Instancehood is declared using the `isa` predicate:

```
(isa Fido Dog)
(isa Dog Collection)
```

Collections can be declared to be subcollections, or "specs" of other collections using the `genls` predicate as a subclassing relation as follows:

```
(genls Fruit PlantPart)
```

which means that every instance of `Fruit` is also an instance of `PlantPart`.

The FOLifier creates a unique unary predicate for each collection, e.g. `Fruit(x), PlantPart(x)`. This has the potential to make resolution-style inference easier because there is less choice of literals to resolve on due to greater variety in the predicate position. Furthermore, it is assumed that the heuristics of theorem provers are generally optimized for type hierarchies being expressed as unary predicates.

Given this, the `genls` assertion shown earlier can simply be translated as:

$$\forall x[\texttt{Fruit}(x) \Rightarrow \texttt{PlantPart}(x)]$$

Exceptions to the above technique are the *reified collection-denoting terms*, e.g. `(JuvenileFn Human)` which represents the collection of all young humans. (Note that a collection-denoting function need not necessarily denote a subcollection of its argument, for example `(FruitFn AppleTree)` does not denote a subcollection of `AppleTree`.) In our FOLification, we ignore the functional nature of the term and simply create a new predicate by concatenating the names of the function and its argument(s).

3.3 Predicate and Function Quantification

Allowing predicates to be arguments to other predicates and functions often enables more compact representations in the Cyc KB.

For example the meta-predicate `TransitivePred(x)` can be defined as follows:

`TransitivePred(P)` $\Rightarrow$
$\forall x \forall y \forall z \texttt{P}(x, y) \wedge \texttt{P}(y, z) \Rightarrow \texttt{P}(x, z)$

A more complicated example is

```
relationAllExists(pred, type1, type2)
```

which states that for every instance of `type1`, `x`, there must be some instance of `type2`, `y` s.t. `pred(x,y)`. For example `relationAllExists(has,Dog,Nose)` states that every dog has a nose.

In ResearchCyc terminology, pseudo-higher-order constructs such as the above are called *rule macro predicates* and it is usually straightforward to convert these into FOL by replacing the literal with a first-order axiom schema, as in the `TransitivePred` example.

Another case in which predicate/function quantification can be translated into FOL is with `argIsa` assertions that are used to constrain the value of arguments to a predicate. For example,

```
argIsa(performedBy, 1, Action)
```

states that the first argument of a `performedBy` assertion must be an instance of `Action`. This can be translated into FOL as follows:

$$\forall x \forall y [\texttt{performedBy}(x, y) \Rightarrow \texttt{Action}(x)]$$

3.4 Variable Arity Predicates and Functions

It is natural to regard many predicates and functions as having variable arities, e.g. `TheSet` which extensionally enumerates a set. In some cases, the arities can be bounded by a small number, e.g. the Cyc function `Meter` can take either 1 or 2 arguments. `(Meter 1)` denotes "one meter", and `(Meter 1 2)` denotes "between one and two meters". In those cases, we can translate into FOL by suffixing the name of the predicate or function with the arity with which it appears in the assertion i.e.

In many cases, for example `TheSet`, the arity is fundamentally unbounded. In these cases, no straightforward translation seems to exist. An exception is the `different` predicate which, when applied to n arguments, can be translated into $O(n^2)$ inequality relations.

3.5 Exceptions

Exceptions to assertions in Cyc can be declared using the `exceptFor` and `exceptWhen` predicates, e.g.

```
(exceptFor Taiwan-RepublicOfChina
(implies (isa ?X ChineseProvince)
(geopoliticalSubdivision China-
PeoplesRepublic ?X)))
```

Exception assertions like the above are used to give Cyc non-monotonic behavior.

When translating an exception assertion into FOL, the FOLifier could take the exception conditions and add them

FOL untranslatability reason	# of Assertions
UNBOUNDED ARITY FUNCTION	77756
META-SENTENCE	54209
META-VARIABLE	18923
VARIABLE ARITY FUNCTION	14187
FUNCTION ARG CONSTRAINT	7481
HOOK	6407
COLLECTION QUANTIFICATION	2263
UNBOUNDED ARITY PREDICATE	1554
FUNCTION QUANTIFICATION	1072
PREDICATE QUANTIFICATION	766
ILL-FORMED	458
INTER-MT	295
UNEXPECTED	247
SEQUENCE-VAR	206
NON-COLLECTION	132
NON-FUNCTION	114
NON-PREDICATE	62
VARIABLE ARITY PREDICATE	0

Table 1: ResearchCyc Translation Statistics

as negated literals in the antecedent of the excepted rule. In the general case this is more difficult to handle. Also, there can be problems when the assertion and the exception appear in different microtheories.

Currently, our FOLifier does not handle exceptions.

3.6 Translation Statistics

Table 1 gives a breakdown of how each assertion in the ResearchCyc KB was handled by our FOLifier. Overall 960,327 (out of 1,074,484) assertions were translated successfully. For the others, we list the reasons why a translation was not possible and the number of assertions for which that reason applies.

UNBOUNDED ARITY PREDICATE and UNBOUNDED ARITY FUNCTION indicate that the assertion has a predicate or function with variable arity, and that there is no upper bound on the maximum arity, as described in Section 3.4.

VARIABLE ARITY PREDICATE and VARIABLE ARITY FUNCTION indicate that the assertion has a predicate or function with variable arity, but there is an upper bound on the maximum arity. Many of these arise from Cyc's representation of scalar intervals. These are in principle translatable as described in Section 3.4, but our initial FOLifier implementation only handled fixed-arity relations.

META-SENTENCE indicates that a sentence occurs as a term (e.g. within a modal, or an exception).

META-VARIABLE indicates that the assertion references a meta-variable, which is an intrinsically higher-order language feature.

FUNCTION ARG CONSTRAINT indicates a constraint on the argument of a function. Our FOLifier implementation does not currently handle these.

HOOK indicates that the assertion contains a "hook" into some procedural code.

FUNCTION QUANTIFICATION, COLLECTION QUANTIFICATION and PREDICATE QUANTIFICATION indicate that the assertion quantified over a function, a collection (which translates into a predicate), or a predicate that was not a rule macro predicate.

INTER-MT indicates that the assertion involved some kind of inter-contextual reasoning between microtheories using the `ist` predicate that could not be translated into FOL.

SEQUENCE-VAR indicates that the assertion used a variable that stands for a sequence of terms rather than a single term. These are used in Cyc to write higher-order rules that quantify over predicates that could have varying arities.

NON-COLLECTION, NON-FUNCTION, NON-PREDICATE, and UNEXPECTED were cases which our FOLifier was unable to handle. Some of these were due to `Lambda` and `Kappa`, since function-denoting functions and predicate-denoting functions are inherently non-firstorderizable.

ILL-FORMED illustrates some kind of syntactic or semantic problem with the Cyc assertion itself.

4 The Cyc Inference Engine

The Cyc Inference Engine is a higher-order theorem prover optimized for the Cyc Knowledge Base. It differs from most automated theorem provers in several fundamental ways. Cyc's inference engine is designed to reason over a very large common-sense KB. It is incomplete, non-monotonic, and handles context reasoning natively. It is highly modular; in addition to modules implementing inference rules, it has modules implementing meta-reasoning and meta-meta-reasoning. It can dynamically handle theory revisions requiring only limited, local knowledge recompilation, and truth maintenance is always on.

4.1 Large Common-Sense KB

ResearchCyc contains over a million axioms, so the inference engine's datastructures and heuristics are designed to handle hundreds of thousands of constants and tens of thousands of predicates efficiently. Furthermore, the knowledge in Cyc's KB is *common-sense* knowledge. Common-sense knowledge is more often used in relatively shallow, "needle in a haystack" types of proofs than in deep mathematics-style proofs. Common-sense knowledge is more often used for constructive proofs than proofs by contradiction.

These factors have significantly influenced the design of Cyc's inference engine in ways quite different from most other automated theorem provers, which are often optimized for very different types of knowledge, such as formal verification, mathematics, or a single specific domain.

4.2 Incompleteness

Common-sense knowledge is tightly interconnected, and that fact combined with a very large KB make for intractably large branching factors in inference. Cyc's inference engine is heuristically guided and resource-bounded; it trades completeness for efficiency. At some point, the size and density

of a KB become so large that any complete inference algorithm would be so slow as to be pragmatically useless. At that point, it becomes wise to give up completeness in favor of efficiency.

4.3 Highly Modular

Cyc, like most state of the art theorem provers, has a sophisticated set of heuristics used to rank potential inferences in a preference order. Also like most theorem provers, Cyc considers applying a set of inference rules at each inference step.

However, most theorem provers have a small handful of inference rules, usually including hyperresolution and paramodulation. In contrast, Cyc employs a "pandemonium" model in which hundreds to thousands of "Heuristic-Level (HL) modules", each of which implements an inference rule, check whether they are applicable to the given subproblem and make a bid to solve it. HL modules can be dynamically added or removed from the system without requiring any sort of retuning or recompilation.

4.4 Higher-Order

Some higher-order features are efficiently implemented via HL modules. For example, Cyc has an HL module that applies only to transitive predicates. When invoked, it performs a graph walk over the indexing structures of the KB, which is considerably more efficient than performing unrestricted resolution.

By representing the knowledge that a given predicate P is transitive as `TransitivePred(P)` rather than as a rule, Cyc gains parsimony in knowledge representation as well as efficiency in inference. If there are N transitive predicates in the universe of discourse, Cyc can represent this knowledge in $N + 1$ clauses: 1 clause per predicate plus 1 very general higher-order rule. In a straightforward FOL representation, $3N$ clauses would be required: 1 three-clause rule to express the transitivity of each predicate.

Many other higher-order features are handled analogously, with corresponding gains in both inference effiency and KR parsimony.

4.5 Contexts

Reasoning within a hierarchy of contexts is built into the innermost loop of Cyc's inference engine. Cyc maintains a dynamic contextual scope which can differ for each literal of each subproblem of an inference. Each axiom considered for use in a proof is first checked for relevance to the contextual scope. In the general case, this is done via efficient graph walking of the context hierarchy (a directed graph, not necessarily acyclic), and for the common cases, the computation is cached. Hence, Cyc can handle contextual and inter-contextual reasoning very efficiently.

5 Experimental Results (Preliminary)

Here we present the results of our preliminary experiments on performing inference with the first-orderized ResearchCyc KB.

In the last 20 years a number of efficient first-order theorem provers have been developed for doing sound and complete reasoning in first-order logic. The most common and successful design for these theorem provers is a resolution-style saturation strategy with sophisticated heuristics to determine clause and literal weights.

The gold standard for these theorem provers has been the TPTP (Thousands of Problems for Theorem Provers) library, which contains first-order encodings of problems from various domains as diverse as lattice theory to circuit verification. The challenge in most of these problems has been to find "deep inferences": starting from a relatively small set of axioms (small as compared to, say, Cyc's common-sense ontology) and returning a proof that requires a large number of inference steps.

Common-sense queries, on the other hand, typically require "shallow inferences", i.e. proofs that use a very small percentage of the entire KB and do not have many steps. The problem is to choose at each stage the correct line of inference to pursue out of all the possibilities. It has been observed (Reif & Schellhorn 1997) that goal-directed strategies are the only practical methods to use when the numbers of axioms are more than a few hundred.

Our ResearchCyc FOLification was motivated in part by the opportunity to experiment with first-order theorem proving strategies on the ResearchCyc KB and compare those results with the performance of Cyc's own inference engine.

Unfortunately we ran into a number of practical difficulties in using first-order theorem provers for our common-sense KB, a task for which they were not optimized. At the time of publication, we were unable to obtain the kind of extensive results that would enable us to draw confident conclusions. We instead present some initial work and our speculations.

With the exception of E, none of the theorem provers we tried were able to load more than 20% of the ResearchCyc KB without failing due to memory errors. In most cases, this was due to internal data structure limitations in the theorem-proving programs, which we are trying to fix with the help of the maintainers of these programs. Meanwhile, we have performed a few experiments with a subset of the ResearchCyc KB, as a feasibility study.

These experiments were performed as follows: We selected a set of 8 common-sense queries from an existing test corpus of queries for the ResearchCyc KB. This fixed set of queries was chosen before running any experiments. We selected these 8 in particular because they represent a diverse collection of different types of common-sense queries representable in ResearchCyc. None of these 8 queries turned out to be inherently dependent on higher-order reasoning; all of them were answerable by pure first-order reasoning on the translated KB.

We then randomly generated a subset of the FOLified ResearchCyc KB[2] containing less than 10% of the total number of axioms (about 100,000 out of 1,250,000) and manually ensured that the all the axioms needed to prove all the

[2] In the case of SPASS however, because of licensing issues, this was done on the OpenCyc KB, with virtually identical results.

Query	VAMPIRE (sec.)	SPASS (sec.)	E (sec.)	Cyc (sec.)
`(isa isa Individual)`	18.1	16.4	26	0.006
`(implies (and (subOrganizations ?Z ?X) (hasMembers ?X ?Y)) (hasMembers ?Z ?Y))`	3.4	5.2	12	2.5
`(typePrimaryFunction Bathtub TakingABath deviceUsed)`	1.6	2.8	9	0.02
`(typeBehaviorIncapable Doughnut Talking doneBy)`	43.4	29.1	132	0.03
`(implies (and (parts ?X ?Y) (parts ?Y ?Z)) (parts ?X ?Z))`	8.7	12.2	23	0.6
`(disjointWith Baseball-Ball Cube)`	176.5	343.0	1491	0.02
`(disjointWith HumanInfant Doctor-Medical)`	847.6	1239.1	2934	0.04
`(implies (and (isa ?CUP CoffeeCup) (isa ?COFFEE Coffee-Hot) (in-ContOpen ?COFFEE ?CUP)) (orientation ?CUP RightSideUp))`	218.4	462.5	574	0.7

Table 2: Inference Experiments

queries were present. (This manual step explains why we could only experiment with 8 queries.)

Finally we ran the theorem provers VAMPIRE, SPASS and E (which collectively represent the state of the art among ATP systems) on the KB + negated query. For SPASS, we had to first convert our KB into its native DFG format (Weidenbach 1999). The results are shown in Table 2. We have also shown the performance of the Cyc inference engine on these queries. The experiments were performed on an Athlon 2800 with 512 MB of RAM.

As an example of the completely different approaches to inference taken by the ATP systems and the Cyc inference engine, consider their behavior on the sixth query, `(disjointWith Baseball-Ball Cube)`.

Cyc solves the query in the following way: The inference engine delegates the problem to an HL module that can efficiently solve `disjointWith` queries. First it walks up the `genls` graph for `Cube`, marking each node as a goal. Then it walks up the `genls` graph for `Baseball-Ball`, and for each more general collection, it walks across all explicit `disjointWith` assertions to check whether it hits a goal node. In this particular example, the inference path is:

1. Mark `Cube`, `RectangularParallelepiped`, `Parallelepiped`, `Polyhedron`, etc. as goals
2. Iterate through each genl G of `Baseball-Ball`, e.g. `Ball`.
3. Iterate through each of G's asserted disjoins D, e.g. `Polyhedron`.
4. Check to see if D is a goal node.

This simplified walkthrough glosses over many details, not the least of which is the fact that microtheory reasoning is built into the innermost loop of this algorithm; on each traversal of each link in the `genls` and `disjointWith` graphs, Cyc does another graph walk of the `genlMt` graph, the graph in which Cyc's hierarchy of contexts is stored, to ensure that the link is relevant to the current context.

In comparison, the first-order approach is simple and uniform: Each `genls` and `disjointWith` assertion is translated into a rule such as $\forall x \forall y Baseball\text{-}Ball(x) \Rightarrow Ball(x)$ or $\neg(Ball(x) \wedge Polyhedron(x))$. Negating the query, $\exists y(Baseball\text{-}Ball(y) \wedge Cube(y))$ and running a resolution-style proof search will then yield a contradiction and hence a positive answer.

5.1 Interpretation

A few caveats have to be made before conclusions can be drawn from these experiments. In order to obtain these results, a number of simplifications were required in our experimental procedure. First, we had to scale down the size of the KB as described above in order to load it into all the theorem provers, whereas Cyc's inference engine was run on an unmodified KB (over 10 times the size of that used by the theorem provers). Furthermore, for all the queries except the last one, we 'flattened' the query context for the theorem provers, i.e. the query was asked in the BaseKB and all its supporting axioms were placed in the BaseKB. This in effect meant that first-order inference required one less unit propagation step on the `mtVisible` literal. The last query was not flattened, but was asked in the context of `TerrestrialFrameOfReferenceMt`. This may be a factor in its relative slowness. Cyc's inference engine did not flatten any of its queries or any of its KB. In all the deviations we made in the testing conditions for the FOL theorem provers versus the ResearchCyc inference engine, we ensured that we erred on the side of the theorem provers, i.e. the changes sped up their running times.

The immediate observation is that Cyc's performance is orders of magnitude better on almost all the queries. We believe that this validates our hypothesis that a specialized inference engine working natively with compact higher-order

representations and special-purpose reasoning modules will perform better on common-sense queries than general first-order theorem proving strategies. However, much work remains to be done before this claim can be asserted with confidence.

The only query for which Cyc's result is within an order of magnitude of any of the other results is the second one, the $hasMembers$ query. This is because Cyc solves the query by hypothesizing terms for $?X$, $?Y$, and $?Z$ and performing forward inference on them before moving on to query-focused backward inference. 2.3 of the 2.5 seconds are spent in forward inference, which proves to be unnecessary for this particular query. This suggests further optimizations to Cyc; e.g. if the forward inference were done lazily rather than eagerly, Cyc could answer the question in 0.2 seconds rather than 2.5 seconds.

An alternative explanation for the difference in results is that our tests with most theorem provers were not performed under ideal settings. We did not have a chance to experiment extensively with the various parameter settings and heuristics that the programs offered; we used a uniform Set-Of-Support resolution setting for all of them. It could be that with, for example, an improved clause/literal weighting strategy, these theorem provers would become competitive with Cyc.

Another possible explanation for the difference in results is the inherent incompleteness of the FOLification procedure, as discussed in section 3. Perhaps it omits some axioms that are key to proving the queries efficiently in FOL, or perhaps it could translate to a form more amenable to other theorem provers. We hypothesize that this is not the case, and we plan to test our hypothesis by gathering input from the authors of other theorem provers, refining our FOLification procedure, and rerunning the experiment.

It should be noted, however, that most state of the art theorem provers have been optimized for mathematical problems requiring deep inference, such as those in the TPTP library. Cyc would almost certainly perform very poorly on those problems. Conversely, the FOLified ResearchCyc KB can be viewed as a new challenge problem for Automated Theorem Proving. With our results as a baseline, can theorem-proving strategies be found that improve performance on common-sense queries to match those of a specialized inference engine like Cyc?

The apparent poor performance of E relative to the other two theorem provers is surprising, especially given the fact that E performed very well in recent ATP competitions (Pelletier, Sutcliffe, & Suttner 2002) and that E was the only program that could load the entire ResearchCyc KB. This might be explained by the fact that E does not directly do Set-Of-Support resolution, but instead uses a clause weighting strategy that simulates it.

6 Conclusions and Future Work

The ResearchCyc FOLification tool was motivated by a number of goals. First, we aimed to measure the inherent higher-orderness of the ResearchCyc KB, which is a good indication of how much higher-orderness common-sense KBs in general can be expected to require. Going by sheer number, we were able to translate around 90% of the axioms. However, the other 10% of axioms that remained untranslated may form the core of Cyc's ontology. To verify this, experiments need to be set up with a large corpus of queries from ResearchCyc's test suite. The percentage of queries that can be answered by both ResearchCyc and any ATP system gives a measure of how much of the information content of the KB is actually translated, and how much of the inferential power actually carries over to FOL.

The second goal was to measure the performance of the Cyc inference engine against state of the art theorem provers. Our tentative conclusions are given in Section 5.1. The benefits of such a comparison could be mutual. First, if it is found that on certain classes of queries, there are theorem-proving strategies that work better than Cyc's inference engine, then we would like to explore the possibility of incorporating these methods into ResearchCyc, perhaps as HL modules. We are encouraged in this direction by the success of SUMO (Niles & Pease 2001) in using VAMPIRE.

Conversely, common-sense queries over large KBs of this scale represent an entirely new class of problems that have not been studied extensively in the automated reasoning community, mainly due to a lack of problem sets. The FOLified ResearchCyc KB could help kickstart research into this area. We have discussed with the maintainers of the TPTP problem library the possibility of generating a suite of common-sense benchmark problems from the ResearchCyc KB.

As for our experiments, a lot of work remains to be done before a completely fair comparison can be made between ResearchCyc and the ATP systems. Once we are able to perform experiments on the entire FOLified KB, then we can automatically run them on hundreds of queries in the ResearchCyc Test Corpus. The statistics gathered can then be used to draw definite conclusions about both representability and efficiency in reasoning.

It is quite likely that on many of the problems that modern theorem provers excel at (for example a theorem from group theory), Cyc's inference engine would perform poorly or not return an answer at all. Given the radically different design decisions of Cyc's inference engine versus those of most other theorem provers, it is not surprising that they have very different performance characteristics on different types of problems. Now that ResearchCyc is available to the academic community, these design decisions can finally be put to the proving ground. Is it possible to design a first-order theorem prover that can efficiently answer realistic common-sense queries over a large KB? Could such a theorem prover already exist, modulo a (perhaps more sophisticated) HOL to FOL translation mechanism? The experimental results presented in this paper are a first step toward answering these questions, and we challenge and encourage others to continue this work.

7 Acknowledgements

We would like to thank a number of people for their past and ongoing help. The first author would like to thank Eyal Amir for his encouragement and advice. We would like to thank the designers and maintainers of the theorem provers we

used in our experiments, for helping us use their software on our KB: Stephen Schulz (E), Andrei Voronkov (VAMPIRE) and Thomas Hillenbrand (SPASS). We would especially like to thank Geoff Sutcliffe for his help with the TPTP translation and for running the model checking software Paradox on OpenCyc. This work was funded in part by ARDA's NIMD program.

References

Boolos, G. 1984. To be is to be the value of a variable (or to be some values of some variables). *Journal of Philosophy* 81:430–449.

Buvac, S. 1996. Quantificational logic of context. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*.

Giunchiglia, F. 1993. Contextual reasoning. *Epistemologia* XVI:345–364.

Lenat, D., and Guha, R. 1990. *Building Large Knowledge Based Systems*. Reading, Mass.: Addison Wesley.

McCarthy, J. 1986. Notes on formalizing contexts. In Kehler, T., and Rosenschein, S., eds., *Proceedings of the Fifth National Conference on Artificial Intelligence*, 555–560. Los Altos, California: Morgan Kaufmann.

Niles, I., and Pease, A. 2001. Towards a standard upper ontology. In Welty, C., and Smith, B., eds., *Proceedings of the 2nd International Conference on Formal Ontology in Information Systems(FOIS-2001)*.

Pelletier, F.; Sutcliffe, G.; and Suttner, C. 2002. The Development of CASC. *AI Communications* 15(2-3):79–90.

Reif, W., and Schellhorn, G. 1997. Theorem proving in large theories. In Bonacina, M. P., and Furbach, U., eds., *Int. Workshop on First-Order Theorem Proving, FTP'97*, 119–124. Johannes Kepler Universität, Linz (Austria).

Riazanov, A. 2003. *Implementing an Efficient Theorem Prover*. Ph.D. Dissertation, University of Manchester.

Schulz, S. 2002. E - a brainiac theorem prover. *Journal of AI Communications* 15(2):111–126.

Sutcliffe, G., and Suttner, C. 1998. The TPTP Problem Library: CNF Release v1.2.1. *Journal of Automated Reasoning* 21(2):177–203.

Weidenbach, C. 1999. *Handbook of Automated Reasoning*. Elsevier. chapter SPASS: Combining superposition, Sorts and Splitting.

An Application-Oriented Context Pre-fetch Method for Improving Inference Performance in Ontology-based Context Management

Jaeho Lee, Insuk Park, Dongman Lee, and Soon J. Hyun

School of Engineering
Information and Communications University
103-6 Munji, Yuseong, Daejeon, Korea
{leejaeho, ispark, dlee, shyun}@icu.ac.kr

Abstract

Ontology-based context models are widely used in a ubiquitous computing environment. Among many benefits such as acquisition of conceptual context through inference, context sharing, and context reusing, the ontology-based context model enables context-aware applications to use conceptual contexts which cannot be acquired by sensors. However, inferencing causes processing delay and it becomes a major obstacle to context-aware applications. The delay becomes longer as the size of the contexts managed by the context management system increases. In this paper, we propose a method for reducing the size of context database to speed up the inferencing. We extend the query-tree method to determine relevant contexts required to answer specific queries from applications in static time. By introducing context types into a query-tree, the proposed scheme filters more relevant contexts out of a query-tree and inference is performed faster without loss of the benefits of ontology.

1. Introduction

An application is considered context-aware if it adapts to a user's or its own context such as location, state, and so on. There are several examples of context-aware applications. Among them is Teleport System (Dey et al. 2001) in which a user's desktop environment follows the user as he or she moves from one workstation to another. Another example is the Navigation system (Baus et al. 2002) which dynamically displays the navigation information according to the speed of traveling in order to help a user's attention. To facilitate the development of context-aware applications, a context model is required to manage such functions as storing, searching, and sharing contexts that change dynamically.

There have been many research efforts on context modeling such as application-oriented model (Dey et al. 2001; Kindberg et al. 2000), graphical model (Henricksen et al. 2002), and ontology-based model (Gu et al. 2004; Chen et al. 2004). The ontology-based context model is widely used because of its advantages of sharing and reusing knowledge, and especially, of inferring a conceptual context which cannot be acquired by the data gathered from sensors.

Several context management systems have proposed ontology-based context models for the rapid and reliable development of context-aware applications (Wang et al. 2004; Lee et al. 2004; Ranganathan and Campbell 2003; Khedr and Karmouch 2004). The main roles of a context management system are to collect contexts from sensors, to infer conceptual contexts, and to deliver an appropriate context to applications. Inference, however, is time- and resource- consuming. The processing time for inference increases proportionally to the size of contexts as well as the number of applied rules. As a ubiquitous computing environment becomes more intelligent, the size of contexts grows accordingly. This makes it difficult to achieve the inference result in a timely manner. Although most context management systems with the ontology-based context model recognize the problem, none of them provides solutions to it.

In this paper, we propose a context pre-fetch method to reduce the size of context database to be loaded in a working memory [1] to speed up inference. Our experiences in developing context-aware applications show that only a subset of the whole contexts is used in inferencing, e.g., less than 40 contexts out of 2,000 contexts. Therefore, we argue that irrelevant contexts can be filtered out from the whole contexts in a working memory. We extend the query-tree method (Levy et al. 1997) to identify the contexts required to answer specific queries of applications in static time. We adopt constraint predicates of the query-tree method to restrict the contexts and pre-fetch the contexts relevant to applications' queries. We classify contexts into three type categories; sensed, deduced, and defined context. With constraint predicates, the context types are also used as irrelevance claims to determine whether a certain context is required for them or not. Experimental results show that the proposed scheme allows the size of contexts in a working memory and the processing time for inference to be maintained smaller without loss of the benefits of

[1] Working memory is a memory space used for inference.

reasoning than the query-tree method. It certainly helps ontology-based context management to be scalable in a resource-limited ubiquitous computing environment.

The rest of this paper is organized as follows. In chapter 2, we show related work. In chapter 3, our approach is explained in detail. In chapter 4, we describe an implementation, followed by the evaluation in chapter 5. Finally in chapter 6, conclusion is provided and future work is discussed.

2. Related Work

2.1. Ontology-Based Context Model

The Context Broker Architecture (CoBrA) has developed an ontology-oriented context model to make easy knowledge-sharing across distributed systems (Chen et al. 2004). They have used the F-OWL inference engine to get a conceptual context from raw contexts (Zou et al. 2004). The F-OWL inference engine has features to enhance reasoning performance. The engine introduces the Tabling method to reduce the processing delay required for rule-based inference. The Tabling is used in various inference engines such as the Jena2 (A Semantic Web Framework for Java) (Carroll et al. 2004). According to it, once it is proved that a subject-property-object triple is in the target ontology model, the triple is added to an appropriate table. Even though a few queries at first may take time to get results, the next queries get quick responses. However, when update of the model such as data assertion or retraction takes place, the table gets invalidated. Therefore, it is impossible for the context models to take advantage of Tabling as context changes frequently at runtime.

The Semantic Space has also developed an ontology-based formal context model to address critical issues including formal context representation, knowledge sharing, and logic-based context reasoning (Wang et al. 2004). The system uses Jena2 (Carroll et al. 2004) to inference the ontology-based contexts. However, similar to CoBrA, performance degrading is a problem when the size of the context increases. They suggest three solutions. First, the time-critical context-aware applications execute without reasoning process. Second, context reasoning is independently performed by resource-rich devices such as the residential gateway. Among them, most active solution is two-level structure, i.e., high-level and domain-specific. By dividing the context into two levels, the size of context used in inference can be reduced because each domain-specific context is managed separately and loaded dynamically when context-aware applications move into the domain. However, in this case, not the entire context is used by context-aware applications in a domain, and the aforementioned performance degrading problem of the logic-based reasoning still remains when the size of a domain becomes large. We also take the two-level structure to design our context ontology. Furthermore, the proposed method in this paper focuses on reducing the domain-specific contexts since all domain-specific contexts are not relevant to a specific query, or a context-aware application. In this way, our method makes the system scales well even though the size of a domain becomes large.

2.2. Methods to Speed up Inference

We have examined efforts to speed up reasoning inside inference engines. Some of the ontology inference engines nowadays are Jena2 (Carroll et al. 2004), Racer (Racer), FaCT (FaCT), Hoolet (Hoolet), and Triple (Triple). Jena2 is a Java framework for writing Semantic Web applications. Jena2 provides inference for both the RDF and OWL semantics. However, the response time of reasoning increases along with the number of triples because Jena2's memory-based Graph model simply has been implemented as the triple pattern (S, P, O) matches by iterating over the Graph. Although the reasoning engines such as Racer, FaCT, Hoolet, and Triple are well-developed, none of them scales well when dealing with the OWL test case *wine ontology* (http://www.w3.org/TR/owl-guide/wine.owl) (Zou et al. 2004).

Alon Y. Levy, et al. proposed a method for search space pruning to speed up inference using the query-tree is shown (Levy et al. 1997). The query-tree is a powerful static-analyzing tool for determining knowledge base containing rules and finite structure that encodes all derivations of a given set of queries. Using the method, we can select rules and ground facts used in deriving answers to the queries. We adopt and extend the query-tree method to determine relevant contexts required to answer specific queries given from applications at the time of initializing the application.

3. Context Pre-fetch Method

3.1. Design Considerations

In the ontology-based context model, the speed of inference can be improved by reducing two factors; (a) the number of rules and (b) the size of context database (Gu et al. 2004). Rules are divided into two types. One is rules for ontology reasoning such as `subClassOf`, `Symmetric`, and `Transitive` semantics as shown in Figure 1. Ontology reasoning is responsible for checking logical requirements which include class consistency, class subsumption, and instance checking (Gu et al. 2004). Since it does not make sense losing any semantics of ontology for speeding up the inference, the rules for ontology reasoning are kept in a working memory at runtime. The other is user-defined rules for generating conceptual contexts. These rules are needed only when context-aware applications ask for those conceptual contexts. We assume that user-defined rules required for an application's operation are given by the application. Thus, we cannot reduce the number of both ontology and user-defined rules in order for an application to work correctly even though the small number of rules makes the inference time shorter. Therefore, we focus on

reducing the size of context database for speeding up inference.

Reducing the size of context database is a process in which the system reasons only relevant contexts to an application's query and fetches and places them in a separate context database in a working memory. This process is the same as Irrelevance reasoning in AI. The relevant contexts are what the context-aware application's queries need to access to get their results at runtime. We can guess relevant contexts from the context-aware application's queries by decomposing them into primitive contexts, or ground facts. One of techniques to guess relevant contexts is the query-tree method (Levy et al. 1997).

In the query-tree method, a query-tree built from a given query is traversed and irrelevant facts are pruned using *constraint predicates*. Constraint predicates are used to specify restrictions on ground facts. For example, suppose that a query, `AgeOf(x, y)`, is restricted by a constraint predicate, `y<=150`. It means that the ground facts larger than 150 are irrelevant to the query. They are found statically using constraint predicates before the query is requested and excluded when evaluating the query at runtime. In other words, only the ground facts satisfied with `y<=150` are placed in a working memory for the query at runtime. However, in the case of a context-aware application's queries, there are many queries about contexts acquired from sensors. The contexts acquired from sensors do not need to be placed in a working memory until the value of context comes in from sensor since they are meaningless before real values are sensed. It motivates us to devise a method for filtering further contexts which are relevant but useless to applications out of the query-tree. It helps to scale up the ontology-based context management in such a resource-limited ubiquitous computing environment. We introduce context types, which indicate when and how context can get a meaningful value, for further filtering besides constraints predicates. They are described in the next section.

Transitive-Property	(?P rdf:type owl:TransitiveProperty) (?A ?P ?B) (?B ?P ?C) -> (?A ?P ?C)
Symmetric Property	(?P rdf:type owl:SymmetricProperty) (?X ?P ?Y) -> (?Y ?P ?X)
inverseOf Property	(?P owl:inverseOf ?Q) (?X ?P ?Y) -> (?Y ?Q ?X)
Equivalent Property	(?P owl:equivalentClass ?Q) -> (?P rdfs:subClassOf ?Q), (?Q rdfs:subClassOf ?P)
subClassOf	(?A rdfs:subClassOf ?B) (?B rdfs:subClassOf ?C) -> (?A rdfs:subClassOf ?C)

Figure 1. Part of OWL Property Rules

3.2. Context Representation and Categorization

We design context ontology for a home environment using Web Ontology Language (OWL). Context ontology is divided into upper-level and lower-level context ontology similar to the CONON (Gu et al. 2004). An upper-level context defines each class and property, and expresses the relationships and constraints between properties using ontology semantic rules. A lower-level context defines instantiations of domain-specific facts using general concepts and properties of an upper-level one. For example, lower-level contexts both `'Bedroom'` for a home domain and `'Office'` for a business domain are described by an upper-level context, `'Room'` as shown in Figure 2.

A context is encoded as a triple which has a form of (subject, predicate, object) in OWL. While the subject and object are merely physical and logical entities or sensed values, the predicate makes a semantic relation between two entities. For example, the `'hasDevice'` property of `'Bedroom'` in Figure 2 is represented as a form of `'<Bedroom, hasDevice, Bed>'`. In addition, the context of a triple form can be extended to represent a complex context by combining the predicate.

We classify a context into a sensed context, a deduced context, and a defined context like (Gu et al. 2004; Henricksen et al. 2002). Every predicate of a context has a property, `'owl:classifiedAs'` to specify its type. (referred to as the `'Room'` ontology in Figure 2). We use the types of contexts to determine whether a query should be used to build a query-tree for selecting relevant contexts. Other researchers also proposed context categorization but they used it for other purposes, i.e., expressing the quality of a given context, compensating for context imperfection (Gu et al. 2004; Henricksen et al. 2002).

First, a sensed context such as `'person;locatedAt'` is acquired from a sensor at runtime. A sensed context is dependent on a sensor running in an environment. Therefore, a sensed context is meaningless until a sensor corresponding to a given context actually works at runtime. Accordingly, a context query for a sensed context such as `'<?p person;locatedAt Bedroom>'` cannot get an answer before runtime. Second, a deduced context such as `'person;hasStatus'` is acquired only by inference after the sensed context, `'person;locatedAt'` is sensed. To get a deduced context, we define user-defined rules which consist of context queries relevant to other types of contexts. For example, a deduced context such as 'a user's current status is `sleeping`' is obtained only when the current contexts satisfy the sleep-rule[2]. It contains context queries for the current state of a bed, and a person's current location. Among these queries, `'<?p person;locatedAt Bedroom>'` and `'<?d device;hasState ON>'` are the queries for a sensed context obtained at runtime. Accordingly, the sleep-rule cannot generate in static time the deduced context such that a person's current status is 'sleeping'. Finally, a defined context such as `'<TV device;locatedIn Bedroom>'` is defined by a user and

[2] sleep-rule : (?p rdf:type person:Person) (?d rdf:type device:Device) (?p person:locatedAt room:Bedroom) (?d device:locatedIn room:Bedroom) (?d device:hasState ON) -> (?p person:hasStatus status:sleeping).
The rule consists of five queries. Each query has a literal as variable. Among them, (?d device;hasState ON) means what is devices whose hasState value is ON>.

is rarely updated over its lifetime once after it is determined. Therefore, the result of a context query for defined context is always the same whenever the query is evaluated. Consequently, a defined context is only a context that can be acquired before runtime. Thus, we consider context queries for a defined context in a pre-fetch process.

```
<owl:Class rdf:ID="Room">
<rdfs:subClassOf>
   <owl:Restriction>
   <owl:onProperty rdf:resource="#doorState"/>...
<owl:ObjectProperty rdf:about="&room;Brightness">
 <rdfs:domain rdf:resource="#Room"/>
 <rdfs:range rdf:resource="&state;Brightness"/>
 <owl:classifiedAs rdf:resource="&icu;Sensed"/>
</owl:ObjectProperty>  ......
```

(a) Upper-level Context Ontology

```
<Room rdf:ID="Bedroom">
 <hasDevice rdf:resource="&device;Bed"/>  ......
 <Brightness rdf:resource="&state;BedroomBrightness"/>
 </Room>
```

(b) Home Domain Context Ontology

```
<Room rdf:ID="Office">
 <hasDevice rdf:resource="&device;Desk"/>  ......
 <hasDevice rdf:resource="&device;Fax"/>
</Room>
```

(c) Business Domain Context Ontology

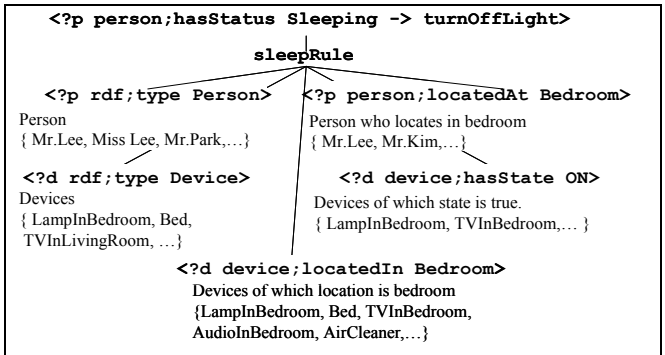
Figure 2. 'Room' Upper-level Context Ontology and 'Bedroom' and 'Office' Lower-level Context Ontology

3.3. Context Pre-fetch

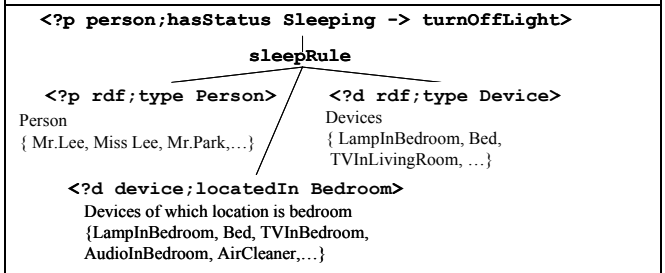
The proposed method uses a pair of memory spaces; (a) a working memory and (b) a pre-processing memory. The working memory is used to perform inference over the contexts that support a context-aware application's operation at runtime. The pre-processing memory is used for the selection of relevant contexts in static time before use. We define 'pre-fetch' as a series of processes for building a query-tree, selecting relevant context in the pre-processing memory, and delivering the selected context into the working memory.

Before the pre-fetch process, we load the upper-level context ontology into the working memory at the initialization time to use for ontology reasoning. Relationships and constraints defined in the upper-level ontology are used to check the consistency of asserted contexts or infer contexts at runtime. For examples, the 'locatedAt' property of the'Person' upper-level context ontology and the 'hasPerson' property of the 'Room' upper-level context ontology are in the 'owl:inverseOf' relationship to each other. In such a case, an assertion of a sensed context in a working memory such as '<Mr. Lee locatedAt Bedroom>' makes the value of the 'hasPerson' property of the 'Bedroom' context ontology 'Mr. Lee'. Thus, the upper-level context ontology has to remain in the working memory during runtime. All context ontology is loaded into the pre-processing memory to examine a query over the whole contexts set.

We prune the contexts irrelevant to an application's operation from the whole contexts in the pre-fetch process. Pruning is done in two steps; (a) pruning by a constraint predicates adopted from the query-tree method (Levy et al. 1997), (b) pruning by predicate types implied by the context category.



(a) Query-tree based on Lighting application's query before filter out queries for sensed context



(b) Final state of query-tree based on Lighting application's query

Figure 3. An Example of Query-Tree

First, we use a constraint predicate to prune the contexts irrelevant to an application's operation. Context queries in a context-aware application are described by the notation of *triple match* which is one of ontology query languages. It returns all statements that match with a template in a form of (subject, predicate, object), where each term is either a constant or a don't-care (Wilkinson et al. 2003). Our context ontology described by using OWL is also encoded in a triple form and allows a property to specify the restrictions of its domain and range. Thus, for a given context query, we can limit the search space of the query from the restrictions described in the context ontology. For example, in a context query, '<?p, locatedAt, ?r>', only a 'Person' type of a context is allowed to be '?p' and only a 'Room' type of a context is '?r' by the 'Person' ontology. We define constraints specified by the restrictions of a predicate described in the upper-level ontology as constraint predicates. We build a query-tree based on the context queries of context-aware applications. Figure 3 (a) shows a query-tree based on a lighting application's query. In Figure 3 (a), '<?p person;hasStatus Sleeping>' query is for deduced context. A query for deduced context can be derived into sub-queries for other

types of a context. Therefore, a given query is divided into sub-queries which are '<?p rdf;type Person>','<?d rdf;type Device>','<?p person;locatedAt Bedroom>','<?d device;locatedIn Bedroom>', and '<?d device;hasState ON>'. The label of each node in Figure 3 (a) represents the constraints specified by the predicate of each triple. Context queries for a context-aware application's operation are derived from building a query-tree. And then, the query-tree helps generate the relevant contexts satisfying the constraint predicates of each node on it. In the query of a lighting application, the number of relevant contexts pruned by the constraint predicates is about 3,000 out of almost 6,000 contexts.

After the first step, we use predicate types to filter out context queries for sensed contexts in a query-tree. As explained in the previous section 3.2, the queries for a sensed context cannot be any answered at the query-tree build time, so they can be filtered out from the query-tree. By filtering out the queries in advance, the number of queries needed to be pre-fetched is further decreased. Accordingly, the pre-fetch processing time is reduced and irrelevant contexts are pruned. Figure 3 (b) shows a query-tree after the first and second steps.

Finally, remaining context queries in a query-tree are evaluated at the pre-processing memory and then, the result of evaluating contexts are delivered into the working memory for further inference.

Figure 4 shows whole procedures of context pre-fetch.

1. Initialize the pre-processing memory and the working memory.
   1.1. Load upper-level and lower-level context ontology into the pre-processing memory.
   1.2. Load upper-level context ontology into the working memory.
2. Build query-trees based on queries of a context-aware application at the time of initializing it.
3. Resolve queries for deduced context into queries for other types of context.
4. Filter out queries for sensed context in the query-tree.
5. Evaluate the remaining queries in the query-tree on the pre-processing memory loaded upper-level and lower-level context ontology.
6. Assert the results into the working memory for the inference engine.

Figure 4. Procedures of Context pre-fetch

4. Implementation

We have implemented the proposed method as part of our ubiquitous computing middleware, Active Surroundings (Lee et al. 2004). We designed a context ontology for a home environment in OWL and used the Jena2 Semantic Web Toolkit for evaluating rules and queries over the context ontology. First, we show a running process of a lighting application in the Active Surroundings without pre-fetch. Describing how a context-aware application runs using the middleware is to provide better understanding on how the pre-fetch method works for the middleware using the ontology-based context model.

A lighting application is depicted conceptually in Figure 5. The operation of the lighting application is very simple such that a light turns off when the user's status is 'sleeping'. In Figure 5, for the lighting application to be performed, it needs to be subscribed to the context management system in advance. The Context Wrapper, of which a concept is introduced in (Dey et al. 2001), transforms a signal from sensors into a form of a context and updates it to the working memory. The Context Aggregator possesses a rule to produce a conceptual context and generates it when the present values of a context satisfy the conditions of the rule. Sleep Aggregator checks whether a user is 'sleeping' or not. Therefore, the aggregator is automatically registered also. Likewise, required by the Sleep Aggregator, the Location Wrapper and the Bed Wrapper are registered automatically as well. Each of the registered aggregators and wrappers keeps a list of the applications and aggregators that use its context. When context changes occur to an aggregator or a wrapper, it notifies to the applications or aggregators in the list. At runtime, the Location Wrapper and the Bed Wrapper obtain contexts about current user location and bed status from the sensors and reflect them to the working memory. Then, the Sleep Aggregator examines the value of location and bed context. It concludes the current status of the user to be 'sleeping' and reflects it to the working memory. Finally, the lighting application examines whether the user's current status is 'sleeping', and turns off the light if conditions are satisfied.

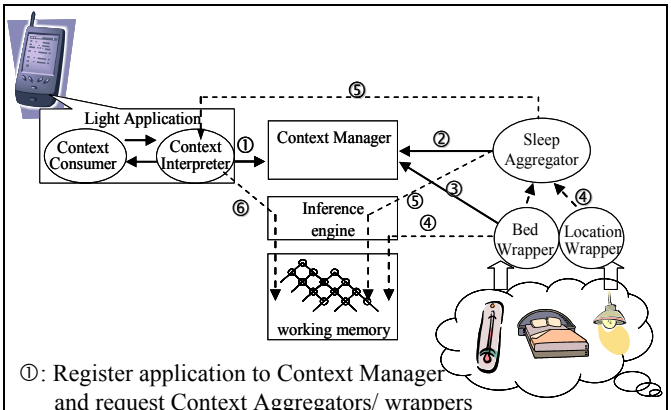
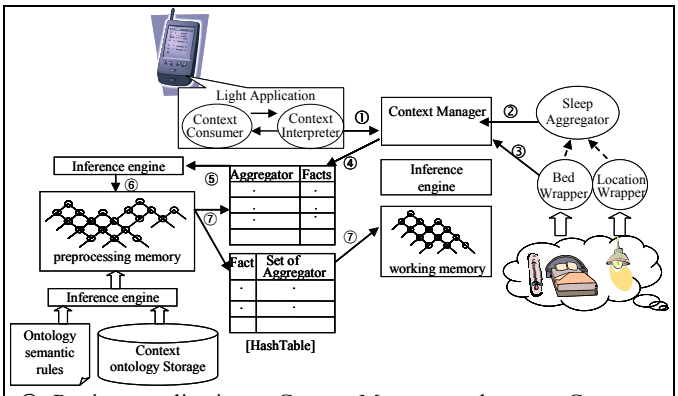


Figure 5. Operation of Context-aware Application without pre-fetch method

As shown in Figure 6, the context management system delivers queries in the registered Context Aggregators and

Context Wrappers to the pre-fetch component at that time. We use two hash-tables to keep the consistency of a context ontology in the working memory. One of the hash-table stores context facts and a Context Aggregator as the hash keys to check whether the Context Aggregator is pre-fetched already. Context facts mean the results after pre-fetching about the Context Aggregator. The other hash-table stores pairs of a set of Context Aggregators and a context fact as the keys to check whether the context fact can be retracted from the working memory at runtime or not. The set of Context Aggregators means a set which consists of Context Aggregators of which pre-fetch result is the context fact. Through the hash-tables, the result set of pre-fetch for a Context Aggregator is stored and managed to support dynamic assertion to and retraction from the working memory. For instance, when a new application is registered at runtime, Context Aggregators needed by the application are checked in the first hash-table if they were pre-fetched already. If they exist in the hash-table, the pre-fetch process is skipped. On the other hand, when a Context Aggregator is unregistered to the context management component, the context facts corresponding to the Context Aggregator are checked using the first hash-table. Then, whether the context facts are still used by other Context Aggregators is checked using the second hash-table. If none of Context Aggregators uses context facts, then they can be retracted safely from the working memory. Figure 6 shows operation process of the lighting application on top of Active Surroundings with the proposed method.



①: Register application to Context Manager and request Context Aggregators/ Wrappers.
②: Register Context Aggregator to Context Manager and request Context Wrappers.
③: Register Context Wrappers to Context Manager
④: Check whether the requested Context Aggregator is pre-fetched already.
⑤: If the Aggregator is not in the hashtable [aggregator-facts], build a query-tree of the query in the ContextAggregator.
⑥: Pre-fetch relevant context using the query-tree.
⑦: Add the result facts to [aggregator-facts] and to [fact-set of aggregator] hashtable, and insert the facts into working memory.

Figure 6. Operation of Context-aware Application with pre-fetch method

5. Evaluation

5.1. Proof of Completeness

We show Theorem 1 that the results of a context-aware application's query both on contexts pruned by the pre-fetch method and on the whole contexts are the same through a proof by contradiction.

Theorem 1: The result of a context-aware application's query on contexts pruned by the pre-fetch method is the same as the one on the whole contexts.

Proof : Let q be a context-aware application's query, and $S_{pre\text{-}fetch}$ be contexts pruned by pre-fetching about q, and $S_{nonpre\text{-}fetch}$ be the whole contexts set, and C_{sensed}, $C_{deduced}$, $C_{defined}$ be the set of sensed contexts, the set of deduced context, and the set of defined context in $S_{nonpre\text{-}fetch}$ respectively, and C_{sensed}', $C_{deduced}'$, $C_{defined}'$ be the set of sensed context, the set of deduced context, and the set of defined context in $S_{pre\text{-}fetch}$ respectively.

Suppose that the result of an application's query on pruned contexts by the pre-fetch method is different from the one on the whole contexts set. Since the types of context are only three, i.e., sensed context, defined context, and deduced context (referred to as Context categorization in section 3.2.), the types of a context query are also three. Thus, the proof is shown in each case separately.

Case 1: Assume that q is a query for the sensed context,

- Results of q on $S_{pre\text{-}fetch} \in C_{sensed}'$ and
  Results of q on $S_{nonpre\text{-}fetch} \in C_{sensed}$
- $C_{sensed}' = C_{sensed}$, because both C_{sensed}' and C_{sensed} are given from sensors at runtime.

Thus, the results of q both on $S_{pre\text{-}fetch}$ and on $S_{nonpre\text{-}fetch}$ are the same, when q is a query for the sensed context.

Case 2: Assume that q is a query for the defined context,

- Results of q on $S_{pre\text{-}fetch} = C_{defined}'$ and
  Results of q on $S_{nonpre\text{-}fetch} = C_{defined}' \subset C_{defined}$, because the results of q are $C_{defined}'$ that is the result evaluated about q in the pre-fetch time.

Thus, the results of q both on $S_{pre\text{-}fetch}$ and on $S_{nonpre\text{-}fetch}$ are the same, when q is a query for the defined context.

Case 3: Assume that q is a query for the deduced context,

- q is divided into sub-queries for the sensed context and defined context in the pre-fetch time,
- Results of q on $S_{pre\text{-}fetch} \in C_{sensed}' \cup C_{defined}'$
- C_{sensed}' and $C_{defined}'$ are proved by Case 1, Case 2.

Thus, the results of q both on $S_{pre\text{-}fetch}$ and on $S_{nonpre\text{-}fetch}$ are the same, when q is a query for the deduced context. In case that q is a query for the deduced context which consists of another deduced context, we can prove it in the same way of the case 3 by decomposing deduced sub-queries recursively until all sub-queries are decomposed into sensed and defined context queries. From the cases 1, 2, and 3, we have a contradiction, which means that the assumption is false. Therefore, it must be true that the result of a context-aware application's query on contexts pruned

by the pre-fetch method is the same as the one on the whole contexts.

5.2. Experimental Result

Experiments were run on a 3.0GHz PC with 1GB of RAM running Windows XP. Our context model in use consists of about 2000 RDF Triples. It can be seen as a small size of context. To show the improvement of the time taken for inference on a large scale context, we extend our context model by defining several domain areas. We also prepare the different type of queries from a simple query to a complex one and practical queries in use on a running system as shown in Figure 7.

Query	Description
Q1 (Simple query)	<?p rdf:type Person> ^ <?p locatedAt Bedroom>
Q2 (Complex query)	<?p rdf:type Person>^<?p gender ?pg>^ <?p birthDate ?pbr>^<?p name ?pn>^ <?p locatedAt ?pr>^<?p hasStatus ?ps>^ <?u rdf:type UserPreference>^ <?u onPerson ?p>^<?u hasWeight ?uw>^ <?u hasService ?us>^<?d rdf:type Device>^ <?d used ?du>^<?d hasService ?ds>^ <?d hasState ?dst>^<?d hasDimLevel ?ddl>^ <?d locatedIn ?r>^<?r rdf:type Room>^ <?r hasPerson ?p>^<?r hasDevice ?d>^ <?r SoundLevel ?rs>^<?r DoorState ?rd>^ <?r Brightness ?rb>
Q3 (WatchTV rule)	<?p rdf:type Person>^<?d rdf:type Device>^ <?p locatedAt ?d>^<?d hasState xsd:true>
Q4 (Sleep rule)	<?p rdf:type Person>^<?d rdf:type Device> <?p locatedAt Bedroom>^ <?d locatedIn Bedroom>^ <?d hasState xsd:true>^ <?l hasDimLevel xsd:0>
Q5 (EnterBedroom rule)	<?p rdf:type Person>^ <?p locatedAt BedroomDoor>

Figure 7. Sample query for evaluation

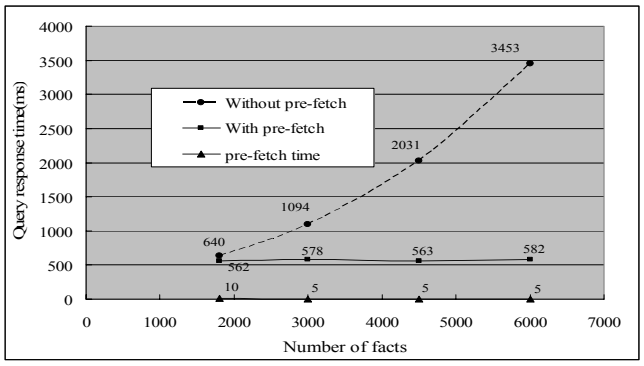


Figure 8. Query response time in simple query

Figure 8 shows the result of the experiment for Q1 (simple query) on context database in the working memory. As described in Figure 7, Q1 query is very plain. Thus, the response time of query is affected strongly by number of context facts in a working memory. Therefore, the query response time also increase as a consequence when the number of facts increases in the case without applying pre-fetch method. While, the graph of the result using the pre-fetch method shows the fixed response time for a query regardless of the increase of the number of facts. The processing time for pre-fetching the relevant context facts is negligible.

Figure 9 shows the result of the experiment for Q2 (complex query) on context database in the working memory. Q2 is a complex query that needs most of context facts stored in the working memory. For the complex query, the response time of the case applying the pre-fetch method increases according to the number of context facts. It is because there are many relevant context facts pre-fetched for the Q2 query. In the case of a very complicated query, many context facts are pre-fetched. However, even the worst case, the response time becomes no longer than the case without pre-fetch because the size of pre-fetched results is not bigger than that of the whole context set in any case.

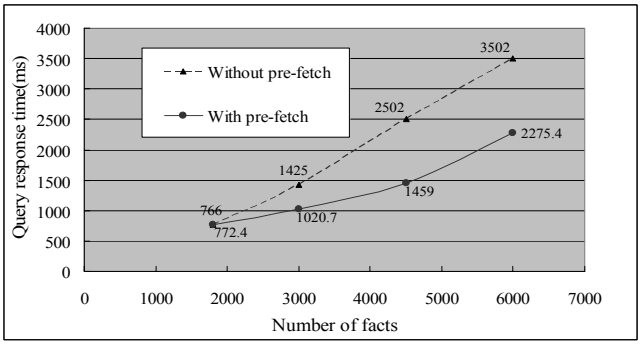


Figure 9. Query response time in complex query

Finally, we test an environment where several realistic applications supporting people's daily life run actually. As shown in Figure 7, three queries, Q3, Q4, and Q5, are used to activate our sample applications. In Figure 10, each graph shows the increase of the response time according to the number of context facts at runtime.

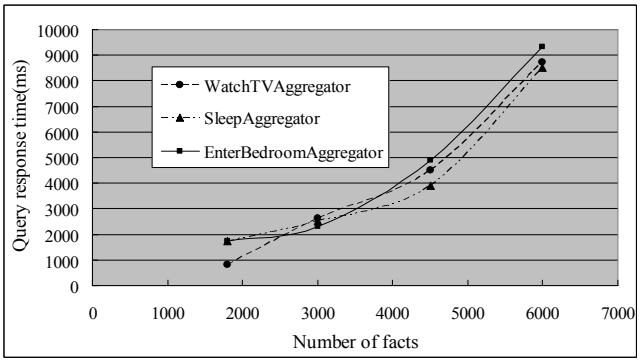


Figure 10. Query response time in Active Surroundings which doesn't apply the pre-fetch method

The results of the experiment with applying the pre-fetch method are shown in Figure 11. The value depicted in

graph is the time which adds the processing time for pre-fetch and the response time of the three queries. As shown in Figure 11, the response time remains constant.

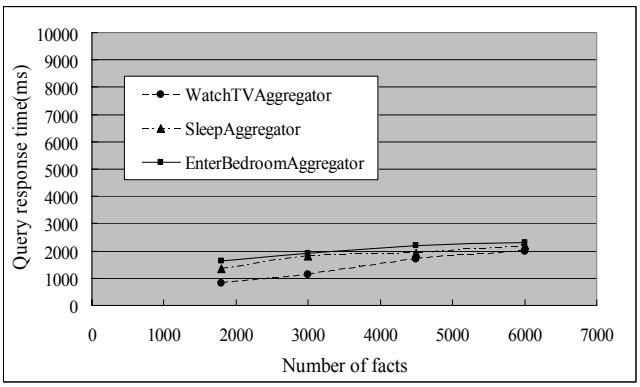


Figure 11. Query response time in Active Surroundings which apply the pre-fetch method

We conclude from the experiments described above that when the number of context facts is less than about 2000, both the result with pre-fetch method and one without pre-fetch method show similar performance. However, if the number of facts is over 2000, the method with the pre-fetch method significantly outperforms the method without the pre-fetch method.

6. Conclusion and Future works

Nowadays, there are number of infrastructures for enabling context-awareness on the basis of ontology-based context model. We consider it will be needed that the module or method which makes an inference process faster over ontology-based context model.

In this paper, we proposed a method to reduce the size of the contexts in working memory by pre-fetching relevant context based on context-aware application's queries. And we apply the method to the existing context management system which uses ontology-based context model, Active Surroundings. By pre-fetching and maintaining relevant context to support context-aware applications into working memory, the inference time on ontology-based context model can be faster than existing method which maintains the whole contexts into working memory. As the result the context-aware applications which use the context generated through inference process can be run quickly comparing with existing method. We currently investigate how to use query optimization techniques together at the pre-fetch time in order to reduce the processing time for pre-fetch.

References

Wang, X.; Dong, J.S.; Chin, C.Y.; Hettiarachchi, S.R.; and Zhang, D. 2004. Semantic Space: an infrastructure for smart spaces. *Pervasive Computing, IEEE. 3(3): 32– 39*.

Levy, A.Y. et al. 1997. Speeding up inferences using relevance reasoning: a formalism and algorithms. *Artificial Intelligence. 97(1-2): 83-136*.

Dey, A.K. et al. 2001. A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications. *Human-Computer Interaction Journal. 16(2-4):97-166*.

Gu, T.; Wang, X.H.; Pung, H.K.; and Zhang,D.Q. 2004. An Ontology-based Context Model in Intelligent Environments. In *Proceedings of Communication Networks and Distributed Systems Modeling and Simulation Conference*.

Henricksen, K. et al. 2002. Modeling Context Information in Pervasive Computing System. *Pervasive 2002, LNCS 2414, 167-180*.

Kindberg, T. et al. 2000. People, places, things: Web presence for the real world. In *Proceedings of IEEE Workshop on Mobile Computing Systems and Applications*.

Chen, H.; Finin, T.; and Joshi, A. 2004. An Ontology for Context-Aware Pervasive Computing Environments. *Special Issue on Ontologies for Distributed Systems, Knowledge Engineering Review*.

Lee, D. et al. 2004. A Group-Aware Middleware for Ubiquitous Computing Environments. In *Proceedings of the 14th International Conference on Artificial Reality and Telexis-tence (ICAT)*.

Carroll, J.J. et al. 2004. Jena: Implementing the Semantic Web Recommendations. *WWW2004*.

Racer. Available online at http://www.racer-systems.com/.

FaCT. *Description Logic (DL) classifier*. Available online at http://www.cs.man.ac.uk/~horrocks/FaCT/.

Hoolet. *OWL-DL Reasoner*. http://owl.man.ac.uk/hoolet

Triple, http://triple.semanticweb.org/

Zou, Y. et al. 2004. F-OWL: an Inference Engine for the Semantic Web. In *Proceedings of the 3rd International Workshop on Formal Approaches to Agent-Based Systems*.

Baus, J. et al. 2002. A resource-adaptive mobile navigation system. In *Proceedings of Intl. Conf. on Intelligent User Interfaces, San Francisco.*

Fensel, D. et al. 2000. Lessons learned from applying AI to the web. *International Journal of Cooperative Information Systems. 9(4):361-382*.

Freeman-Hargis, J. 2003. Rule-based systems and Identification Trees. Available online at http://ai-depot.com/Tutorial/RuleBased.html.

Ranganathan, A. and Campbell, R.H. 2003. An Infrastructure for Context-Awareness based on First Order Logic. *Journal of Personal and Ubiquitous Computing. 7(6) :353-364*.

Khedr, M. and Karmouch, A. 2004. ACAI: Agent-Based Context-aware Infrastructure for Spontaneous Applications. *Journal of Network & Computer Application*.

Wilkinson, K.; Sayers, C.; Kuno, H.A.; Reynolds, D.; and Ding ,L. 2003. Supporting Scalable, Persistent Semantic Web Applications. *IEEE Data Eng. Bull. 26(4): 33-39*.

Contexts as Abstraction of Grouping

Christo Dichev and Darina Dicheva

Winston-Salem State University
601 M.L.K. Jr. Dr., Winston Salem, N.C. 27110, USA
{dichevc, dichevad}@wssu.edu
http://gorams.wssu.edu/faculty/dichevc/
http://www.wssu.edu/~dicheva/

Abstract

This paper presents a framework of *context-centered* digital course libraries founded on the Topic Maps paradigm and used for developing an authoring environment for building such libraries. We explore the idea of using contexts to support more efficient information search. The notion of context is perceived as abstraction of grouping of domain concepts and resources based on the existing semantic relationships between them. The suggested framework implies a layered information structure of the library content consisting of three layers, each capturing a different aspect of the information space - conceptual, resource-related, and contextual. The proposed model of context is used for context representation in the TM4L environment, which enables the creation, maintenance, and use of ontology-aware courseware based on Topic Maps.

1 Introduction

Information search is an old and hard problem in computing. With the growth of the web it is becoming harder. Finding relevant and valid information that meets learners' needs is yet harder. If for example a learner is interested in information related to a Prolog implementation of best-first search, a simple Google search using "best-first", "search", and "Prolog" as keywords results in over 5,190 hits. Moreover, a large part of the references provided in the first few pages assume advanced knowledge of Prolog. Searching information relevant to the "Java threads" topic results in an even worse result – the unmanageable amount of more than 2,240,000 references. Finding good quality web resources poses a major problem for users who have not developed efficient search strategies. People often use the principle of least effort in their information seeking. Following this pattern students frequently use easy accessible, rather than higher quality but less accessible information. In the case of learners searching information to complete a learning task, there is another difficulty - their uncertainty about what kind of resources they need. Learners are also often unaware of the complete context of the task in hand. In such cases they need support in getting oriented in the conceptual structure of the domain of the problem, which will help them in retrieving, evaluating, comprehending, and memorizing information. An even more valuable support should include means for locating online material customized to the individual users by taking into account their interests, level of competency in the considered domain, learning styles, etc.

Why finding needed information on the Web is hard? Regardless of the quality of stored information, it is useless unless it can be indexed and efficiently searched. Conventional search engines can help in identifying entities of interest but they fail in determining the underlying concepts or the relationships between these entities. The main problem with the web and current technology is that it is impossible to semantically relate and compare resources. For example, current search engines are not able to interpret and react adequately to requests such as: *Find another document with more technical details than the current page.* They are not capable to refer to the current page and use its characteristics to guide the search based on transitive dependencies. There is no general-purpose search engine that can answer questions such as: *Find the latest article on the topic (of the current page),* or *Show me a paper that is more detailed than this one*, or *Show me a tutorial that is less formal than this one.*

There is a large amount of high quality learning resources on the web already and they should be made more accessible to users. In this paper we explore the idea of using contexts to support more efficient information search. We propose to define contexts as abstraction of clusters of domain concepts and resources based on the existing relationships between them. This is related to our previous work on contexts as well as on the development of a framework of concept-based digital course libraries [3], [2]. The framework is based on using the new Semantic Web technology Topic Maps [10], [13]. The paper is organized as follows. We first outline our general framework, more details of which can be found in [3]. Then we discuss the use of Topic maps for its implementation. Next we propose our approach for incorporating contexts in this framework. Finally, we discuss the proposed contexts' implementation and use in the TM4L environment.

2 Framework of Concept-based Digital Course Libraries

We have developed a framework of concept-based digital course libraries based on using conceptual structures representing subject domain ontologies for classification of the library content. The classification involves linking learning objects (content) to relevant ontology terms (concepts), i.e. using the ontological structure to *index* the library content. The use of subject ontologies that provide shared agreement on the concepts meaning also allows for ontology-based merging of digital repositories. The main components of the architecture are the information repository, information authoring module, and information retrieval module.

2.1 Library Repository

We propose a layered structure of the library repository consisting of three layers capturing different aspects of the information space (see Fig. 1):

- *Semantic layer*, containing a conceptual model of the knowledge domain in terms of *key concepts* and *relationships* among them.
- *Resource layer*, containing a collection of diverse information resources associated with the specific knowledge domain.
- *Context layer*, containing specifications of different views (contexts) on the library resources depending on a particular goal, type of users, etc., by associating components from the other two layers.

Semantic Layer. The introduction of a separate semantic layer that represents the domain ontological conceptualization allows using it from one side as a subject knowledge *directory* that enables natural and intuitive concept-based content browsing, and from another as a *resource* item relevant to learners' goals. The latter allows for exploration of the ontological structure of the subject domain independently of the information resources, which can help learners to improve their overall understanding of the domain.

Resource layer. The resource layer contains *internal* and *external* learning objects. Internal resources are pieces of information about a concept, such as annotations, definitions, characterizations or short descriptions, stored locally in the library. External resources can be any addressable objects referenced by their URI. By using external learning objects from available collections of standardized (LOM [9], Dublin Core) learning objects the need and efforts to create them will be eliminated.

Context Layer. The separation of the semantic layer from the information repository makes it possible to define different semantic structures over the same collection of learning resources or different collections of learning resources connected to the same semantic structure. A context captures a particular view on the learning resources by preserving the relevant semantic relations among them and filtering out the irrelevant. By maintaining a collection of appropriate contexts in the context layer, it is possible to categorize thematically the learning resources, reflecting multiple semantically customized views that correspond to different situations, user goals, communities of learners, etc. Contexts enable users to access the same resources based on navigational strategies in conceptual spaces appropriate to their current needs.

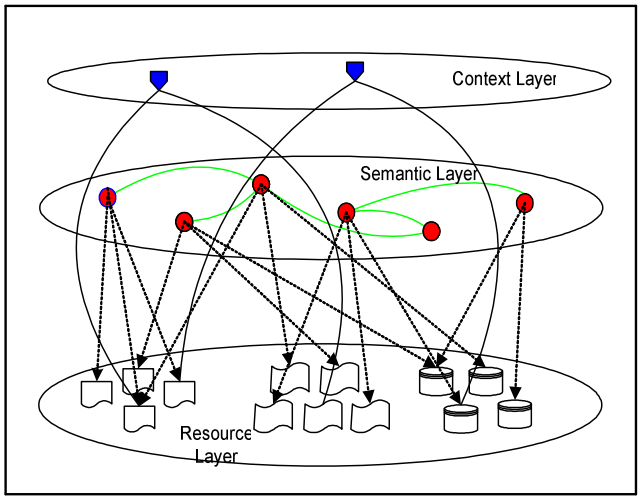


Fig. 1. The layered structure of an information repository.

2.2 Library implementation

We have chosen to implement the proposed general framework of digital course libraries by using the emerging ISO standard Topic Maps [1]. Topic Maps (TM) are appropriate for our goals since they enable users to navigate and access the documents they need in an organized manner, rather than browsing through hyperlinks that are generally unstructured and often misleading.

The main topic maps components are *topics, associations, and occurrences* [10]. The *topics* represent the subjects, i.e. the things, which are in the application domain. They can have zero or more *topic types* and *names* (a base name and possibly variants for use in specific contexts). *An association* represents a relationship between topics. Associations have *types* and define *roles* of the participating topics. *Occurrences* instantiate topics to one or more relevant information resources. The *scope* feature defines the extent of validity of an assertion: the context in which a name or an occurrence is assigned to a given topic, or in which topics are related through associations. An

important concept in TM is this of *identity*. Two topics are the same if both have the same name in the same scope or both refer to the same *subject indicator*. The topics and all their characteristics could be *merged* if this condition holds.

It is clear now that the semantic layer in our framework can be implemented as a collection of associated topics. An important aspect of the topic maps associations is that they can exist despite the absence of occurrences linked to them. Further on, the resource layer can be implemented straightforwardly by defining topic occurrences. The question is how to implement the context layer of the framework in topic maps terms? A quick straightforward answer would be to use the Topic maps *scoping*.

In the TM standard a *scope* is a set of *themes (of validity)*. Themes can be defined and applied to TM objects. The standard allows scoping of topic names, resources, and associations. This is useful for information filtering in Topic Maps Viewers. Obviously a scope can be used to present a context or a perspective however this is a rather static view.

Independently of the standard we propose using topic map associations to represent *context as grouping*. Topic map associations can be interpreted as statements relating topics. For instance, in the case of educational applications, it is possible to express the statement that a given concept is represented using a particular teaching method (e.g. *tutorial, definition, example*, etc.) in the form: topic X *is represented by* tutorial Y. Similarly, associations such as SWI-Prolog *is instance of* Prolog, Prolog *is based on* Resolution, Computation *is part of* Prolog, Prolog *is related to* Horn-Clause Logic, Prolog *uses* Backtracking, make the topic Prolog pertinent to the related topics. Obviously, association types combined with role types enable meaningful grouping of topics, which we call *context.*

Formally context can be defined as a collection of statements that are true in a model. In less formal perspective, context can be interpreted as the things, which surround, and give meaning to something else. The statement "Snow is white" is meaningful if we talk about New Year in Alaska, but has no meaning in terms of CPU scheduling. We can view contexts as a means of grouping facts relevant to a particular situation. Grouping and classification of objects is a human invention to simplify communication. For our purpose we take a restricted model of this view of context, namely, as a grouping of topics based on their relations to a given topic. Translated in topic maps terminology a context is a collection of associations related to a common topic selected to represent and name the context. Technically, this is a nested topic map drawn around a topic chosen to name the context.

We outline our view on context representation in topic maps in more details in the following section.

3 Using Context in Topics Maps

The notion of context includes two aspects that are addressed in the research on modeling contexts. Some authors [8], [11] interpret context as a set of facts describing a particular situation from a specific point of view. Another approach taken for example by [7] is based on the intuition that reasoning is always "local", i.e. it involves only a small subset of what an agent actually knows; this subset is what determines the context of reasoning. However there is no standard way to specify contexts of assertions. Topic maps can be used to model both aspects. The contextual support for organizing (and locating) learning content described above can be interpreted as modeling viewpoints.

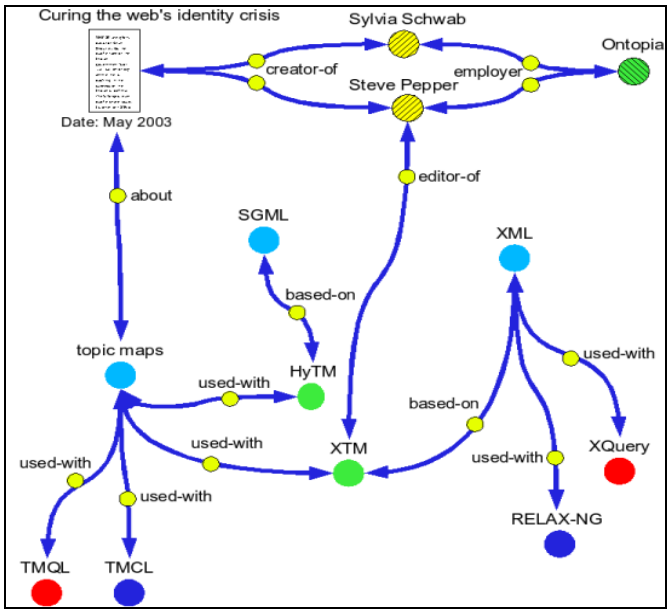


Fig. 2. An excerpt from the Ontopia Topic Map.

The topic maps framework provides support for modeling of context which matches our intuition, namely, that the context is an abstraction of *grouping related information*. Consider the topic *XTM* in the example in Fig. 2 borrowed from Garshol [6]. The objects (facts) that we would consider relevant to this topic are the statements "*XTM is based on XML*", "*XTM is used with Topic Maps*" and "*Steve Paper is editor of XTM*". This collection of facts expresses a clustering of statements. The sticking point of the clustering is the topic "*XTM*". Intuitively these statements make the topics *XML*, *Topic Maps* and *Steve Paper* relevant to *XTM*. The statement that *Steve Paper* is employed by *Ontopia* is less relevant in terms of the current topic (*XTM*) and the fact that "*Puccini is born in Lucca*" would typically be considered as irrelevant. Thus in TM terms we can define context as a grouping of a set of topics clustered around a given topic and therefore considered relevant to that topic. To make the model more

coherent it should account for boundaries that separate one context from another.

Most works related to formalizing context are centered around the so called "box model", where "Each box has its own laws and draws a sort of boundary between what is *in* and what is *out*" [7], [8], [11]. The problem with this approach is that we have to predefine all potentially needed "boxes" in order to use them. The world is too unpredictable to foresee the complete set of contexts that might be needed. Rather than preparing a set of static boxes we suggest to use a TM model that allows shifting boundaries of the context dynamically based on the current topic. We propose to interpret context as a collection of topics surrounding a given topic (denoting the context) and intended to localize the search and the inference within an area of *relevant* topics. The proposed interpretation allows us to introduce a measure of relevancy.

3.1 Topic centered contexts

The starting point is our view of context as a collection of topics that surround, and give meaning to some other topics. The interpretation of what are the surrounding topics is relative. At one point a topic can be a part of the surrounding collection and in another point it might be viewed as surrounded by some other topics giving meaning to it. The relationships are at the heart of semantics, lending meaning to concepts and to resources linked to them.

We begin with the basic assumptions underlying the proposed contextual framework.

- Each context is a collection of topics related to a certain topic of the TM that plays a role of a *focus or center* of the context.
- The central topic is unique and can be used to name the context.
- All semantically related topics identify regions formed by the topics directly or indirectly related to the center of the context.
- The relevance of a topic to the current context is reverse proportional to its distance to the focus of the context.

According to the last assumption the topics of the collection forming a context have no equal status with respect to that context. Their role in the context depends on somewhat spatial properties – the distance to the central topic. The following definitions try to capture the above aspects of the context in more formal terms.

Immediate Propositional Context. An immediate propositional context $c_R(t)$ of topic t is the collection of all associations (statements) $A{:}R_1(t),R_2(t_2),..,R_n(t_n)$ such that the topic t is a role player in that relationship.

$c_R(t) = \{A{:}R_1(t),R_2(t_2),..,R_m(t_m) \mid m \in I,$ t is a role in a relationship of type $A\}$.

Immediate Topical Context. An immediate topical context (or simply immediate context) $c(t) = \{t_i \mid A.R_i(t_i), A \in c_R(t)\}$ of the topic t is the collection of all topics t_i playing a role in the associations $A \in c_R(t)$, that is, the collection of all t_i coordinates of the m-tuples defining the immediate propositional context of the topic t.

Thus the immediate context of topic t is determined by the set of the associations $A_1, A_2,\ldots, A_n$ in which the topic t plays a role (sticking point) and is characterized by the collection of all topics playing a role in A_k, $k = 1, 2, ..,n$.

Context (recursive definition):

1. The immediate contexts of topic t belongs to the context $C(t)$ of topic t.
2. A topic t_1 belongs to the context $C(t)$ of the topic t if there exists topic $x \in C(t)$ and t_1 is an the intermediate context of x, i.e. $t_1 \in c(x)$.

Informally,

Context of topic(t) = All topics directly or indirectly related to t.

Definition. Topic t_1 is related to t_2 if there exists a sequence of associations $A_1, A_2,\ldots,A_n$ such that each pair A_i, A_{i+1} has at least one common role player and t_1 is a role player in A_1 and t_2 is a role player in A_n. The sequence of associations $A_1, A_2, \ldots,A_n$ is called relating sequence of t_1 and t_2 and n is its length.

Definition. The *level of relevancy* of topic t_1 to the central topic t (and thus to the context $C(t)$) is reverse proportional to the length n of the minimal relating sequence $A_1, A_2, \ldots, A_n$ of t_1 and t.

Notice that according to the above definition the level of relevance of a topic t_i to a context $C(t)$ is characterized by the level of relevance to its central topic.

The context $C(t)$ depends on the topic t and is called *current context* when t is the current topic (e.g. the topic being currently visited or observed). Changing the topic t results in changing the context $C(t)$ and thus changing the region of interest.

We can use the following metaphor to illustrate our perception of context: we can view the context like a moving spotlight that throws a strong light on the central topic and to its immediately related topics but only a dim light on the topics that are indirectly related to the central one (where the dimness is proportional to the levels of indirection). Shifting the spotlight changes the set of topics under strong light.

Among the valuable features of this context model is that it provides a mechanism to refer to the *current context*, and use it to identify an area of interest within the TM. This implies that searching for relevant information can be localized into a specified area of interest.

3.2 Relational contexts

Grouping of related objects is a natural way for humans to simplify and comprehend reality. Grouping of related objects can be found in such diverse fields as biology, physics, learning, statistics, economics, psychology, pattern recognition and engineering. We can group places, events, actions, spatial events, social events, and many other types of entities, both concrete and abstract, over an enormous range. In e-learning context learning content can be grouped based on taxonomic or partitive relationships. It can be grouped based on students' knowledge levels, rates of progress, interests, or instructional goals. A grouping can be used to differentiate units that are functionally related. (e.g. program - programming language, programming language - compiler, compiler - parser, parser - lexical analyzer etc.). From these observations we can formulate a context as a *relational grouping of topics*.

Relational context. A relational context $v(A)$ from the viewpoint of the association A is the collection of m-tuples $(t_1,t_2,..,t_m)$ playing a role in an association A:

$v(A) = \{(t_1,t_2,..,t_m) \mid t_i , (i = 1,\ldots m, m \in I)$, are roles in a relationship $A:R_1(t),R_2(t_2),..,R_m(t_m)$ of type $A\}$.

The above definition of context was motivated by some practical considerations such as Topic Map authoring in e – learning settings. Typically the TM author is applying some construction techniques to build a topic map (set of abstract topics). The fundamental construction techniques are *partitioning* of topics (top-down reasoning in a part-whole context); *aggregation* of topics (bottom-up reasoning in a part-whole context), *specialization* of topics (top-down reasoning in a class-subclass and class-instance context), correlation of topics (horizontal reasoning in a relevancy context). As a result, the conceptual knowledge about the *domain to be learned* is structured based on a *taxonomical* and a *compositional hierarchy* (using *class-subclass* and *part-whole* relationships) coupled with horizontal *related-to* relationships. To assist TM authors we had to provide functionality supporting different views such as the *taxonomical (class-subclass hierarchy)*, *class-instance hierarchy* and the *compositional hierarchy* of concepts of the learning content. These views are essentially relational viewpoints/contexts of the topic map displaying related topics based on the particular properties of the relations, e.g. transitivity.

3.3 Contexts, viewpoints, ontologies

When learning material does not appear in isolation, *structure* is needed to encompass a set of learning objects in an instructional unit. For example, a particular unit could belong to one of the general granularity levels *Component, Lesson, Module, Course* and *Program*. These levels are interconnected with *part-of* relations in order to build a complete instructional unit comprising these levels. Thereby, a *Component* is a part of a *Lesson*; a *Lesson* is a part of a *Module*; a *Module* is a part of a *Course* and a *Course* is a part of a *Program.* The *part-of (part-whole)* relation is included in the Dublin Core standard named as *IsPartOf*. The intended use is to relate smaller resources to larger resources or collections that already exist in the collection (e.g. in the library).

The XTM standard does not include *part-whole* relations (as it does for the class-instance and class-subclass relations), but it does support sufficient expressive power to capture most of what one may want to represent about part-whole relations. The standard considers such relations application-specific and therefore does not recommend hard-coding them. Instead the Topic Map standard provides a general construct for defining relations. Therefore for relations such as part-whole the user needs to introduce a dedicated association type. Properties such as transitivity are also defined on application level.

Besides the traditional structuring every unit of the learning content can be related with another unit by multiple kinds of relations, such as a *class-subclass* relationship capturing learning domains taxonomic trees, a *prerequisite* relation capturing learning dependency graphs or a *related-to* relation representing correlation.

The ontologies currently used for structuring e-learning content are typically light-weight. Light-weight ontologies are typified by the fact that they are predominantly taxonomies, with very few cross-taxonomical links. Light-weight ontologies are valid choice in many cases as they are easier to understand, easier to build and easier to get consensus upon. Topic maps are seen as lightweight ontologies because they are able to model knowledge in terms of topics, their classes, instances, occurrences, and associations.

The instruction involves two types of knowledge, the subject (discipline) to be learned coupled with instructional knowledge. From the viewpoint of the category of individuals involved in the learning process there are students and instructors. As a result we distinguish two domains: the domain of the *discipline* to be learned and the *instructional* domain, and also two categories of individuals: learners and instructors. This might be viewed as a high level grouping specifying high level contexts. These four contexts identify in turn four types of ontologies:

1. A *domain ontology*, with object classes from the discipline to be learned.
2. An *instructional ontology*, with topics and relations from the domain of pedagogy.
3. Author's ontology capturing the viewpoint of the instructor.
4. Learner's ontology capturing the viewpoint of the learner.

The distinction of the above viewpoints is essential during the design of an e-learning environment. This distinction was one of the guiding principals when we decided to predefine some relations in TM4L. For example,

taxonomy is vital for the conceptualization of the discipline in categories, on different levels of abstraction. In TM4L class-subclass was included as predefined relation to support generalization/specialization classification.

An organized collection of learning content embodies topics related in different ways. Intuitive interface should support abstract grouping of learning resources such as grouping by unit-structure, goals, learning style, learning paths etc. In such cases a representation in conventional hierarchical structures only is typically insufficient. We needed a model for expressing a grouping of topics based on generic relations. The derived goal was a minimal set of generic relations which covers the needs of the intended applications. The advantage of such an approach is that generic relations subsume particular instances that might be impossible to articulate in specific terms.

In Topic Maps, associations define relations between an arbitrary number of topics. As a primary relation for classifying learning content we have selected the *whole-part* relationship known also as *partonomy* (see Fig. 3) Like a taxonomy, a partonomy is a hierarchy, but based on the *part-of* relation rather than on a *kind-of* relation. The reason for picking out partonomy is its important explanatory role in e-learning context [17]. Explaining what a learning unit is about, often involves describing its parts and how are they composed. For example, we may choose to structure learning material on Programming Languages in terms of its components i.e. Syntax, Semantics and Pragmatics. However, the learning units describing the syntax, semantics and pragmatics are part of the Programming Languages unit and not subclasses of it. By emphasizing the compositional structure, the partonomy is closer to the approach normally used for representing learning content. Recent research in education also indicates that the whole-part presentation method is a technique shown to reduce cognitive load and improve learning [17]. For example, Mayer and Chandler's study [16] suggests that studying initially a part (piece by piece) rather than a whole presentation allows the learner to progressively build a coherent mental model of the material without experiencing cognitive overload.

In many application areas the natural model of the domain requires the ability to express knowledge about the *class-subclass* relation. The class-subclass known also as a *is-a* relation allows us to organize objects with similar properties in the domain into classes. The class-subclass relation has received a lot of attention and is well-understood. However the interaction between whole-part and class-subclass relations has not been studied in any detail. Despite their different purposes knowledge base, database, object-oriented and e-learning communities heavily rely on conceptual models which have a lot in common. Inter-relationships such as *is-a*, *part-of*, *similar-to*, etc. are used to define and constrain the interactions of concepts within these models.

Applications that provide multiple views are able to offer users different perspectives on a selected entity. Therefore, in addition to the primary *whole-part* relationship our TM tool contains four other predefined relationship types, including the classic "class-subclass" (see Fig. 4) and "class-instance" extended with "similar to" and "related to" relations. By offering this minimal set of five relation types we support TM authors that experience difficulties in articulating and naming relationships.

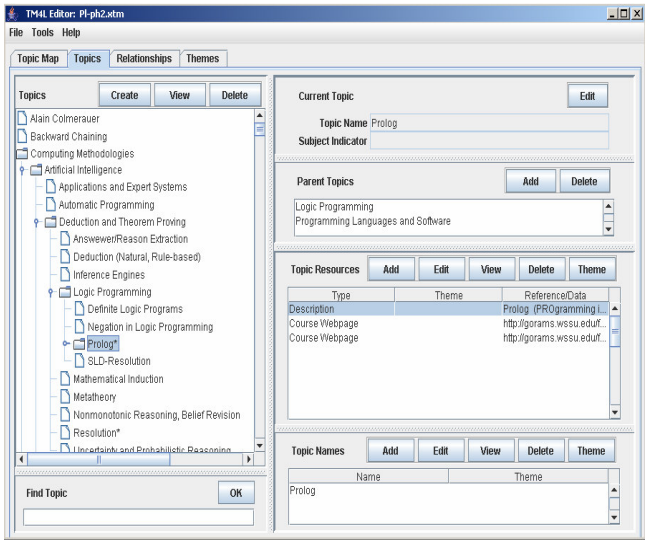


Figure 3 A part-whole view on AI and Prolog.

The proposed set of relations provides also a strategy for organizing the information. It supports a shared way of grouping topics by standardizing the used set of relations. The strategy is based on specifying the set of topics in the domain and the relationships between them in terms of the proposed minimal set. The process of creating the complete contextual structure is incremental; the global TM is a result of growing and interrelating the local structures of immediate contexts.

4 Using Context in TM4L

In the last decade, a number of tools for ontology construction have emerged [18]. Although some currently available ontology editors such as Protégé-2000 (http://protege.stanford.edu/) have plug-ins allowing export to Topic Maps, they do not support essential TM features, which are of significant importance for e-learning applications. The TM4L (Topic Maps For E-learning) environment [6, 7, 8] presented in this paper is intended to complement existing Topic Map editors and visualization tools. It combines two main applications, TM4L Editor and TM4L Viewer. The modeling language of TM4L is based on Topic Maps standard [9]. Two groups of users are targeted by the TM4L design: (i) authors with limited background of ontologies; (ii) learners seeking information support to complete their course tasks. TM4L is currently available as a standalone application. It can be downloaded from: http://www.wssu.edu/iis/nsdl/download.html. The

proposed model of context forms the contextual framework of TM4L which enables the creation, maintenance, and use of ontology-aware courseware.

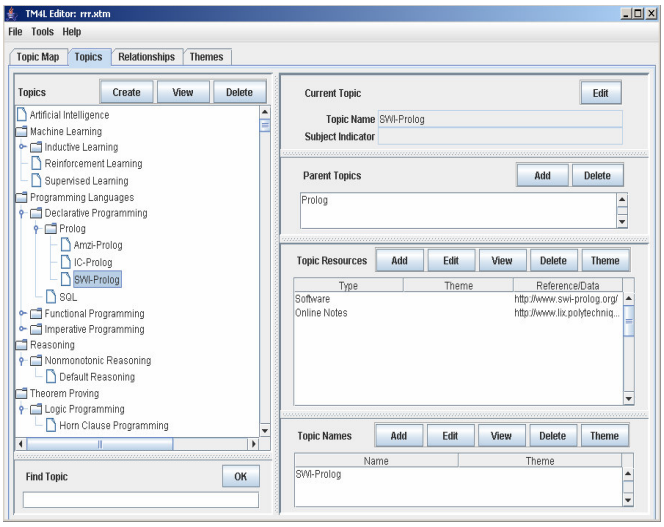


Figure 4 A class-subclass view on AI and Prolog.

There are other software tools employing Topic Maps that can be used to incorporate content into semantically rich data models. One group of tools consists of general Topic Map editors such as the Ontopia Knowledge Suite (http://www.ontopia.net/) - a set of tools for building, maintaining, and using TM-based applications [3] or *Atop* (http://sourceforge.net/projects/atop) - a Topic Map editor written in Java as a NetBeans module using TM4J. With these editors Topic Maps can be interactively built and stored as .XTM (Xml Topic Maps) documents, or in databases. The general TM editors however do not support specific ontological needs such as managing 'whole-part' hierarchy and other types of transitive relations. Besides, they do not support domain specific vocabularies. Instead they tend to use Topic Map-related concepts such as associations, roles, occurrences etc. in their representation. Applications in specific domains such as e-learning require interfaces that supports the particular e-learning objectives coupled with ontology support for classification, navigation and exploration of classes, instances, relations and resources.

As an alternative to conventional authoring systems, TM4L is aimed at facilitating the integration of already existing learning resources on the web. The driving factors and design challenges regarding TM4L interface lie in the following questions: What does the representation mean to learners and authors? Does the representation enable easy detection of the classes of concepts and their relationships? Does it reveal the vocabulary of the domain? Is it immediately apparent which items belong to one or multiple classes, which classes overlap and which don't?

Interfaces that provide multiple contexts are able to offer users different perspectives on a selected entity or on a group of entities. TM4L supports multiple perspectives as the editor and viewer interfaces are context driven. The TM4L Editor provides *Topic centered*, *Relation centered* and *Themes* guided contexts. With TM4L Viewer a topic map can be viewed from six different perspectives: *Subject Topics*, *Relationships*, *Topic Types*, *Relationship Types*, *Resource Types* and *Themes*. In addition, the relational context enables exploration of the e-learning collection in terms of a part-whole tree and a taxonomy tree (see Fig. 3 and Fig 4). Any transitive relation can be mapped to a tree view visualizing particular relational context.

Aiming at reducing the information overload, we have chosen at each navigation step in the TM4L Viewer to display only the topics of *immediate topical context* along with the topics immediately related to them (see Fig 5). In addition, we have chosen not to show the resources associated with the displayed topics in the Graph view, since the visualization becomes too crowded and unclear. Thus the Graph view represents only 'ontology' objects - topics, relationships, roles but not resources. The resources linked to topics can be examined using the Tree or Text views (see the rightmost pane of Fig. 5).

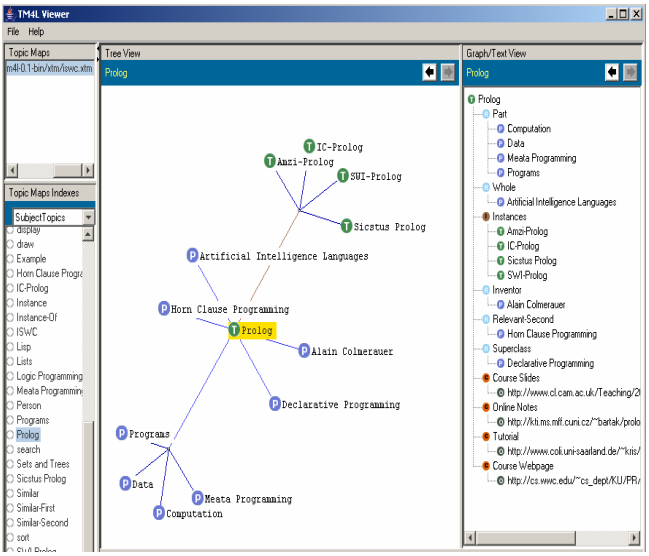


Figure 5. Screenshot from TM4L Viewer with Prolog as a current topic, shown *as a* Declarative Programming Language, *part-of* AI languages, *based on* the Horn Clause Logic and *invented by* Alain Colmerauer. The topics Computation, Data, Programs and Meta Programming are *part of* Prolog while Amzi-Prolog, IC-Prolog, Sicstus-Prolog and SWI-Prolog are *instances of* Prolog.

In addition *Context/theme* filters can be applied to the content shown in the viewer. Every topic characteristic may have a scope, which is specified explicitly, as a set of *themes*. A theme is a topic that is used to limit the validity of a set of topics and relations. The objects that are **not** valid in the specified theme are filtered out. One common use of scopes is to provide localized names for topics.

Name scoping can be used among others for multi-language support. For example, in order to represent the term "Computer science" when browsing a Computer science Topic Map either in English ("Computer science") or in Bulgarian ("Информатика"), the name of the Computer Science topic should be scoped with the themes "English" and "Bulgarian".

5 Conclusion

Efficient information retrieval requires information filtering and search adaptation to the user's current needs, interests, knowledge level, etc. The notion of context is relevant to this issue. In this paper we propose an approach to context modeling in Topic Maps-based educational applications. It is based on the standard Topic Maps support for associations and defines the context as an abstraction of *grouping related information.* The degree of membership of the topics to the context depends on their level of relevancy to the specified topic. This context model provides also a mechanism for referring to the *current context*, and using it to identify a current area of interest within the Topic Map. The second perspective on context is as grouping of *topics that are related to each other in some way*. The notion of context is useful for localizing navigation and search for relevant information within the intended area.

The proposed model of context is utilized in the design of TM4L, an e-learning environment aimed at supporting the development of efficiently searchable, reusable, and interchangeable discipline-specific repositories of learning objects on the web. Providing adequate support for learners to efficiently search for useful web resources is crucial in self-directed learning and presently a problem of high priority in e-learning. We believe that the discussed here approach to context modeling and its implementation in TM4L will contribute to the advancement in that direction by supporting efficient, context-based navigation of educational Topic Maps.

Many different directions for enhancing the TM4L interface are possible. As there are a number of ways to build up the *whole-part* structure of specific learning content, a particular learning unit may be represented by more than one valid partonomic hierarchies. This raises the issue of how to *represent* and *integrate* such multiple hierarchies. In addition to the usual *whole-part* relation, any transitive relation can be used to represent a tree view of the Topic Map.

Acknowledgement

This material is based upon work supported by the National Science Foundation under Grant No. DUE-0333069 "NSDL: Towards Reusable and Shareable Courseware: Topic Maps-Based Digital Libraries" and under Grant No. DUE-0442702 "CCLI-EMD: Topic Maps-based courseware to Support Undergraduate Computer Science Courses".

References

1. Biezunski, M., Bryan, M., Newcomb, S.: ISO/IEC 13250:2000 Topic Maps: Information Technology, www.y12.doe.gov/sgml/sc34/document/0129.pdf. [Last viewed April 5, 2003].
2. Dichev Ch., Dicheva D.: Deriving Context Specific Information on the Web. *Proc. of* the *World Conf. on the WWW and Internet, WebNet'2001*, Orlando, Florida (2001) 296-301
3. Dicheva, D., Dichev, C.: A Framework for Concept-Based Digital Course Libraries, *J. of Interactive Learning Research*, 15(4) (2004) 347-364
4. Dicheva D., Dichev C.: Educational Topic Maps, *3rd International Semantic Web Conference* (ISWC'2004) Poster Abstracts, Hiroshima, Japan (2004) 19-20
5. Dicheva D., Dichev C., Sun, Y., Nao, S.: Authoring Topic Maps-based Digital Course Libraries, *Workshop on Applications of Semantic Web Technologies for Adaptive Educational Hypermedia*, in conjunction with AH 2004, Eindhoven, The Netherlands (2004) 331-337
6. Garshol L.M.: Topicmaps, RDF,DAML, OIL. A comparison. ttp://www.ontopia.net/topicmaps/materials/tmrdfoildaml.html (2002) [Last viewed April 5, 2003]
7. Giunchiglia F. Contextual reasoning, *Epistemologia, Special issue on I Linguaggi e le Macchine* XVI (1993) 345–364
8. Guha, R.: Contexts: a Formalization and some Applications, *Technical Report ACT-CYC-423-91*, MCC, Austin, Texas (1991)
9. IEEE Standard 1484.12.1-2002, Learning Object Metadata (LOM), http://ltsc.ieee.org/wg12/
10. Pepper S.: Ten Theses on Topic Maps and RDF http://www.ontopia.net/ topicmaps/materials/rdf.html (2000) [Last viewed April 5, 2003]
11. McCarthy, J.: Generality in Artificial Intelligence, *Communications of ACM* 30(12) (1987) 1030–1035
12. McGuinness, D.: Ontologies Come of Age. In Fensel, D. et al. (eds) *Spinning the Semantic Web: Bringing the World Wide Web to Its Full Potential*. MIT Press (2002)
13. Park, J., Hunting, S.: *XML Topic Maps: Creating and Using Topic Maps for the Web,* Addison-Wesley (2002)
14. Passin T. B.: Browser bookmark management with Topic Maps, *Proc. Extreme XML*, Canada (2003)
15. Rath, H.H.: Semantic Resource Exploitation with Topic Maps, *Proceedings of the GLDV*-Spring Meeting, (2001) 3-15
16. Mayer, R. E. & Chandler, P. When learning is just a click away: Does simple user interaction foster deeper understanding of multimedia messages? *Journal of Educational Psychology, 93* (2), 2001, 390-397.
17. Price J.L., Catrambone R., Part-Whole Statistics Training: Effects on Learning and Cognitive Load, *CogSci*, Chicago 2004.

Workspaces in the Semantic Web

Shawn R. Wolfe, Richard M. Keller

National Aeronautics and Space Administration
Ames Research Center, Moffett Field, CA 94035-1000
{Shawn.R.Wolfe, Richard.M.Keller}@nasa.gov

Abstract

Due to the recency and relatively limited adoption of Semantic Web technologies, practical issues related to technology scaling have received less attention than foundational issues. Nonetheless, these issues must be addressed if the Semantic Web is to realize its full potential. In particular, we concentrate on the lack of scoping methods that reduce the size of semantic information spaces so they are more efficient to work with and more relevant to an agent's needs. We provide some intuition to motivate the need for such reduced information spaces, called *workspaces*, give a formal definition, and suggest possible methods of deriving them.

Introduction

The technologies of the Semantic Web have yet to achieve the widespread adoption of the World Wide Web. To date, researchers have focused more on foundational issues (e.g., representational formats and capabilities) than on pragmatic issues of scale or efficiency. Ultimately, these practical issues will need to be addressed if the Semantic Web is to gain widespread adoption. In this paper, we focus on one such important issue involving mechanisms for filtering and restricting the set of knowledge statements (e.g., RDF triples) available within a semantic information space, depending on the application context. There are numerous pragmatic reasons why one needs to restrict a semantic space, for example to decrease the search space, limit the scope of reasoning, to improve reasoning efficiency, to reduce information overload, and to customize visual presentations for human users.

As an example, consider an agent searching for "in-plan" providers of a specific medical treatment, as described in Berners-Lee et al.'s influential *Scientific American* article on the Semantic Web (Berners-Lee et al. 2001). Let us presume that there is a semantically marked-up data source that serves as a directory of medical providers. In this case, what steps must be taken for the agent to find the appropriate information? First, it is unlikely that the agent and the directory use the same ontology, so some form of ontology alignment will probably be necessary; this problem has received considerable attention (Kalfoglou and Schorlemmer 2003; Noy 2004). Second, the directory is not likely to be structured in a way that is best suited for the agent's search. The directory may include providers outside the local geographic area, or providers in the wrong specialty area, or it may not make any mention of which providers belong to which insurance plans. In essence, the agent is faced with finding a needle in a haystack; the information it seeks is in the repository, along with a great amount of irrelevant information, and there is no easy way to separate the relevant from the irrelevant. The result is information overload.

One approach to identify the relevant information is to access all potentially relevant information in the directory and use reasoning to restrict the scope. The problem with this approach is one of scale; the more information that is accessed, the more time and computing resources required to store and process the data. Alternatively, if the directory supports searching, the agent may try to scope the space by forming a query that more accurately describes the information request. This approach, too, has its drawbacks. The directory may not support sophisticated queries. Differences in the agent and directory ontologies may require that the query scope be broadened. Finally, the precise query may be very complex, making it difficult to derive and verify that the query will return exactly the desired information.

Information Overload in SemanticOrganizer

We have repeatedly encountered the need to restrict the information space in our work on SemanticOrganizer (Keller et al. 2004), a semantic repository that allows users to store knowledge about work-related items (such as documents, datasets, persons, and other domain-specific concepts) and the interrelationships among these items. SemanticOrganizer has over 500 registered users ranging from occasional users to those who use SemanticOrganizer on a regular basis as the primary storage and retrieval system for their work-related knowledge products. Its single ontology covers a wide variety of domains, from project management to scientific inquiry to accident investigation. SemanticOrganizer has over 400 ontology classes defined, with 45,000 instances of those classes and 150,000 semantic links between these instances.

The SemanticOrganizer system supports various methods of searching and browsing of this information, but as the

size of the repository grows, it produces more dense information displays and voluminous search results – even though much of the information displayed to a user may be irrelevant to their current needs and work context. This problem has forced us to consider methods of restricting the user's information space.

Access permissions, defined on instances within SemanticOrganizer, reduce the amount of information available to a given user but do not fully solve the problem. Because access permissions are intended to prevent unauthorized access rather than access to irrelevant information, they are not an appropriate mechanism for restricting the information space based on relevancy: the problem is not what is accessible, but what is relevant to the user. SemanticOrganizer partially addresses this by allowing users to restrict their semantic space to only instances of certain concepts (i.e., filtering out instances of irrelevant classes). Nonetheless, finer-grained techniques are needed to further reduce information overload -- there may be irrelevant instances of a relevant concept, and irrelevant knowledge statements (i.e., RDF triples) that refer to a relevant instance.

It is possible to view the process of restricting an agent's information space in terms of a series of filtering operations. Consider the following example. Imagine that an accident investigator is browsing information in the SemanticOrganizer repository to orientate herself with a near-miss accident involving equipment failure during an experiment performed in a wind tunnel. Some information would be protected through access permissions and would not be available to the investigator, for instance, the salaries of the employees stationed at the wind tunnel. However, additional information could also be filtered out as irrelevant to *any* investigation, for instance, the investigator's salary. Finally, information that is both relevant to investigations in general and accessible to the investigator, but *not relevant* to the investigation at hand could be filtered out; for instance, water samples taken at the wind tunnel during a previous investigation of a *Legionella pneumophila* outbreak. The information that remains after all the filtering operations are complete is considered part of the investigator's current *workspace*.

The information-scoping problem we have encountered in SemanticOrganizer is a specialization of the more general problem of establishing a common context for communication between two agents, with our specific agents being SemanticOrganizer, on one hand, and a human user, on the other. Our human agents are resource bound just as software agents are, with limits on time and processing power. By establishing a shared context appropriate for the current situation, users can increase their efficiency when "conversing" with SemanticOrganizer. In particular, users will spend less time aligning their mental models to that of SemanticOrganizer. In addition, since the amount of information in a workspace is a subset of the overall information space, users will spend less time sifting through irrelevant information.

Related Work

The problem of restricting an information space to a relevant subset has been the focus of information retrieval (IR), where the problem is usually regarded as retrieving a set of documents from a corpus (see (Salton 1983) for an overview). Typically, the user selects some set of keywords that capture the area of interest, and these keywords are used to query the corpus. The bulk of information retrieval techniques do not make explicit use of semantics, and instead use statistical methods to retrieve relevant documents.

Search queries can be viewed as another way of restricting one's view to a relevant subset. Unlike information retrieval techniques, the search terms must explicitly characterize the subset. Query languages are usually quite expressive, but precise query results often require highly complex queries. As a result, query languages alone are not ideal for adequately scoping the relevant subset. Query languages for the Semantic Web are still evolving, with a variety of languages currently available (Haase et al. 2004). In databases, views defined by queries have been used to limit the scope of subsequent operations. Similarly, variants of RQL have been designed to define a view on a Semantic Web (Maganaraki et al. 2004; Volz et al. 2002).

We have previously suggested a method of restricting a user's view of a semantic repository by choosing a subset of an ontology called an *application module* (Keller et al. 2004). Each application module contains only the classes that are relevant to a particular domain. Knowledge statements that refer to instances of classes not in the application module are filtered out. In addition to filtering, application modules provide some presentation characteristics that allowed users to view instances using their own terminology.

Noy and Musen devised a method for specifying a subset of an ontology through traversal (Noy and Musen 2004). Their focus was primarily on facilitating ontology re-use. Rather than exporting an entire ontology, a user could formulate the relevant portion of the ontology by specifying key concepts and then traversing to related concepts using a traversal directive. Since a procedure can be specified to define the desired subset of the ontology, rather than explicitly choosing the ontology, traversal views offer a greater flexibility and dynamism than the application modules of Keller et al.

Examples of Workspaces

What constitutes an effective workspace will change over time, depending on the intent of the agent. To illustrate the circumstantial nature of workspaces, we present illustrative examples describing the types of workspaces required by an investigator named John during various phases of his work as part of an accident investigation team.

Workspaces Based on the Domain

As John joins the investigation team, his first objective is to familiarize himself with the investigation conducted thus far. John is primarily browsing through the information related to the investigation at this point; he does not have specific information to search for nor does he know what kind of information is available. To support this initial browsing activity, it makes sense to restrict the workspace to only those knowledge statements that apply directly to the investigation at hand. Other information, such as similar investigations at other sites or other investigations at the same site might prove useful to John at a later time, but would currently only make his initial orientation more difficult.

Workspaces Based on a Specific Goal

As John becomes more familiar with the investigation, he naturally proceeds to develop hypotheses, for instance, that poor maintenance procedures led to the failure of a particular machine part. To test his hypothesis, John wishes to restrict his view to only those knowledge statements that relate to the machine of interest and/or maintenance. However, John may choose to consider historical information from other investigations relating to these topics, to find other examples of failures, changes in maintenance procedures, or previous uses of the failed part.

Workspaces Based on Time

Over time, the shape of the investigation changes; new evidence has eliminated some hypotheses and led to new areas of inquiry. To keep abreast of the growing areas of the investigation, John restricts his workspace to include only knowledge statements that have been recently added, for instance statements added during the last week. By doing so, John is directed towards new evidence that would need to be evaluated as well as new hypotheses developed by his co-investigators. Older knowledge statements are no less true, but are no longer novel and therefore of less interest.

Workspaces Based on Task

Finally, as the investigation wraps up, John is tasked with developing a report of the investigation's findings and recommendations. John needs to consider information from all phases of the investigation now, not just the most recently added. However, he is less interested in the details of supporting evidence than in the proven hypotheses, and has no interest at all in the disproven hypotheses. Though John is primarily interested in the current investigation, he wants to bring information from other investigations into his workspace, for example if they discussed findings related to the investigation at hand.

These examples support our viewpoint that the notion of an "appropriate" workspace within SemanticOrganizer is a highly situated notion; the subset of knowledge statements that are relevant to the user at any given point in time depends on the user's work context.

Workspace Definition

Having developed our intuition about workspaces, we now present a more formal definition, illustrated in Figure 1. A workspace is defined with respect to two agents, one a source of information (an information-providing agent: IPA), and the other a requestor of information (an information-requesting agent: IRA).

Let KS_{IPA} be the set of knowledge statements held true by the IPA.

Let $P_{IRA} \subseteq KS_{IPA}$ be the subset of statements that the information-providing agent chooses to publish to the information-requesting agent.

Let $R_{IRA} \subseteq KS_{IPA}$ be the subset of statements that fit some notion of relevancy held by IRA.

Let $C_{IRA} \subseteq KS_{IPA}$ be the subset of statements that can be mapped into the vocabulary used by the IRA. (We assume that there is a partial mapping from statements in the IPA's vocabulary to statements in the IRA's vocabulary – an ontology alignment.) C_{IRA} constitutes the subset of the IPA's statement that the IRA can understand.

With respect to a given IPA, a workspace, W, is defined for a given IRA as follows:

$$W = P_{IRA} \cap R_{IRA} \cap C_{IRA}$$

The workspace for the information-requesting agent is thus defined as the subset of the information-provider's knowledge that the agent is allowed to see, that it can understand, and that is relevant.

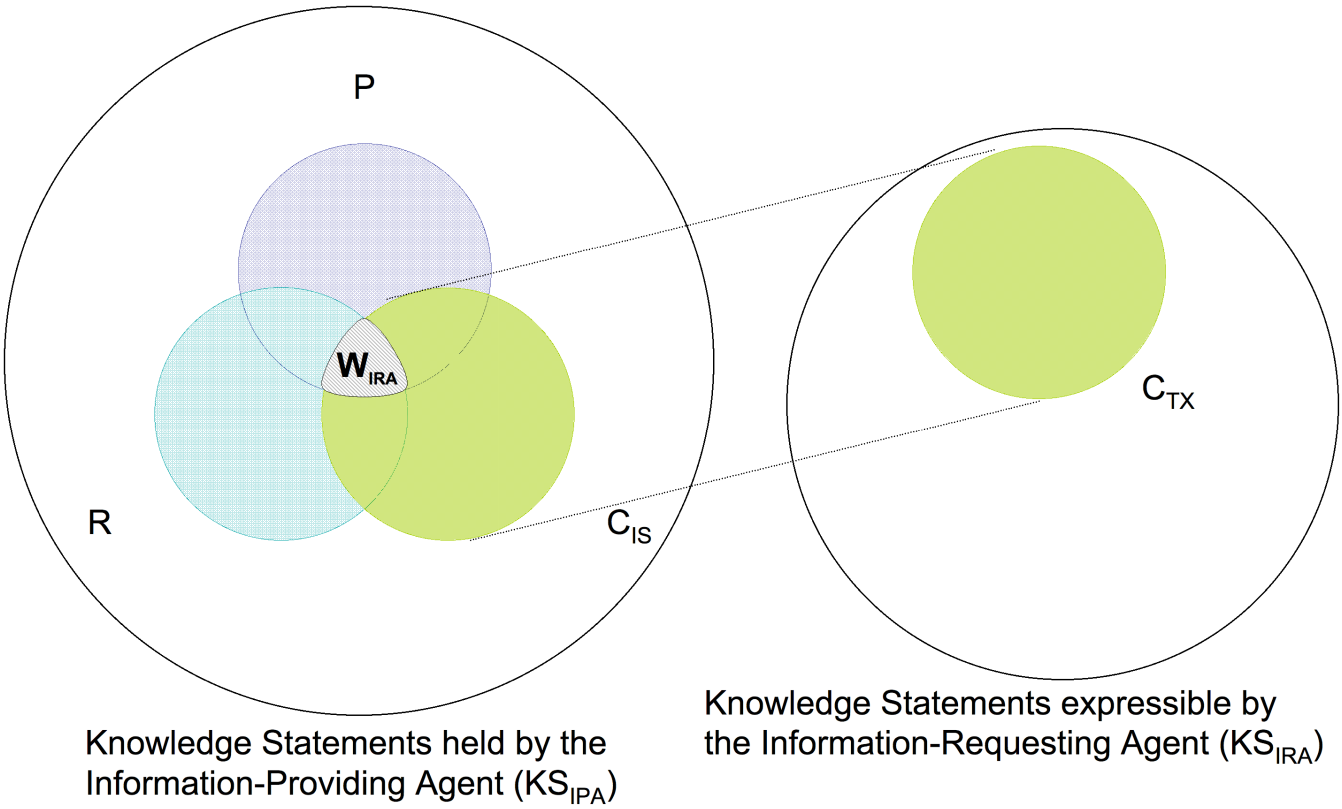


Figure 1: A graphical depiction of a workspace, W, defined for an Information-Requesting Agent (IRA) querying an Information-Providing Agent (IPA). At left is the set of knowledge statements (KS) held true by the IPA; at right is the set of knowledge statements expressible by the IRA. W is defined by the intersection of three subsets of statements held by the IPA: P is the set of statements that the IPA has published to the IRA; R is the set of statements that are relevant to the IRA; and C_{IS} is the set of statements that have a mapping into the vocabulary understood by the IRA

Deriving Workspaces

To derive a workspace, all three of its component subsets must be known. We will presume that the information provider already knows what knowledge statements it is willing to divulge to the information requester, i.e., that it already knows what information it must keep private. Deciding what statements can be translated to the information requester's ontology necessarily involves ontology alignment, another hard problem unto itself that is an area of active research. Within the SemanticOrganizer system, the need to align these ontologies was obviated by *application bundles*, in which ontology specialists customize the master ontology based on the information requester's vocabulary. In what follows, we will concentrate on how to define the third subset – the subset of knowledge statements (R_{IRA}) that fit some notion of relevancy for the information requester. We present three ways to define or derive this *relevant subset*, with each method varying with respect to the amount of semantic interpretation required.

Derivation Via Explicit Selection

The simplest, most obvious method is to manually select the relevant subset of statements, for example by a human knowledge engineer familiar with the agent's context of usage. Manual selection results in the highest quality definition of the relevant subset, but requires the most effort. This method is justified if the manual labor can be amortized over many uses by one or more information requester. For instance, once a workspace is defined for a particular investigation, it could be shared by all the investigators. On the other hand, this method represents no overall reduction of effort if the workspace is used once or infrequently. As with any subset selection method, additions to the overall set of knowledge statements KS_{IPA} would require updating of the relevant subset; since this method is manual, updating can be a significant concern, depending on the frequency of updates.

Derivation Via Description

An alternative to manual selection of relevant knowledge statements is to declaratively describe the relevant subset in

terms of a formal language. The description represents an abstraction of the relevant subset and should require less manual effort to construct than the explicit selection. In contrast to the explicit method above, as knowledge statements are added to KS_{IPA}, the existing description would be used to make the selections, requiring no further effort. This method requires less work than the manual method, but produces a relevant subset that contains a higher number of both irrelevant knowledge statement (false positives) and missing relevant knowledge statements (false negatives).

Derivation Via Ontology-Neutral Learning Methods

Finally, learning methods that use ontology-neutral approaches could be used to drastically reduce the amount of effort required for an agent to define the relevant subset. Such approaches are based on either structural properties of the information space, such as graph connectivity, or meta-concepts and relationships that are relevant across ontologies (for instance, utilizing subsumption or identity relationships, but not domain specific relations). These learning techniques would require limited input if at all – possibly a few training examples. The use of limited amounts of input and lack of domain knowledge will generally result in less accurate results than the previous two more knowledge-intensive methods. Nonetheless, due to the amount of labor involved in manually choosing the relevant subset or describing the relevant subset, such automated methods offer a useful alternative when lower quality subsets are acceptable.

A Simple Experiment

In order to start exploring the space of domain-independent learning approaches, we turned again to the investigation domain of SemanticOrganizer. Four mishap investigations have been supported in SemanticOrganizer (Carvalho et al. 2005): the Columbia shuttle, CONTOUR probe, HELIOS autonomous aircraft, and Canard Rotor Wing (CRW) investigations. Much of the information in these investigations is disjoint, since they occurred at different times, as part of different missions, and involved nearly completely disjoint mission teams. Moreover, most of the common information that could have been included within several investigations was instead (re-)defined separately as part of each new investigation. Therefore, there were very few common *instances* among the investigations. There were a few links crossing between instances included in different investigations, though not many.

Experiment Setup

We considered the case of a single user who has access to information in several investigations, but needs to restrict his view to the subset of information relevant to a single investigation. To define a gold standard for evaluating the formation of the relevant subset, we accessed the accounts of other users who each had involvement in only a single investigation. We used the access permissions of each other user to define the relevant subset of instances for their investigation. In order to simplify the experiment, we focused only on identifying relevant *instances* rather than considering the more numerous relevant *knowledge statements*.

Our goal is to derive these relevant subsets of instances automatically – in this case to derive each subset of instances relevant to a specific investigation. Using the information available to us in SemanticOrganizer, we devised the following experiment. First, we took the union of all the instances and links available to the aforementioned user from all four accounts- this constitutes the items accessible across investigations. Second, to restrict the area to only the domain of investigations, we filtered out all information that was not part of the domain of discourse of investigations (for instance, some information on the ontology itself was represented). Finally, we created a simple algorithm to group the instances into clusters around each investigation.

The Algorithm

Our algorithm takes as input a network of nodes and edges (e.g., an RDF graph), already filtered by permissions and an area of discourse, and *focal instances* that define relevant subsets of instances. Each focal instance is the starting point for a cluster; in our experiment we had four such focal instances, namely each instance of the *Investigation* class. The algorithm produces as output one subset of instances for each focal instance. These subsets may overlap, and the union of these subsets may not include all instances from the original graph. We used the shortest path through the network from an instance to each focal instance as a simple heuristic for deriving the subsets. Each instance was placed within the cluster of the focal instance to which it was closest; if it was equally close to more than one focal instance, it was put in the cluster of each such focal instance. We present the pseudocode of this algorithm below:

For every focal instance F

  Define $S_F = \{\}$

For every instance n in G

  Let C be the set of focal instances closest to n

  For every focal instance F in C

    Add n to S_F

Return: All sets S_F corresponding to each focal instance F

Our intuition was that this algorithm should perform well on this particular task. However, the network was connected, with a path existing from every node to every

other node, so it was possible that the algorithm would not perform well at all.

Experimental Results

On this particular experiment, the algorithm outperformed our expectations. We evaluated the quality of the derived subsets of instances in terms of the information retrieval measures of recall, precision and F-measure (Van Rijsbergen 1979) (Table 1).

	Size of Correct Subset	Size of Derived Subset	Recall	Precision	F-Measure
CRW	349	336	0.82	0.85	0.83
Columbia	4299	4212	0.97	0.998	0.98
CONTOUR	1033	992	0.96	0.998	0.98
Helios	1461	1444	0.99	0.999	0.99

Table 1. Evaluation of derived subsets for each investigation.

Despite these extremely high outcome measures, the conclusions that we can draw from this experiment are very limited. The domain was clearly well-suited to the algorithm's shortest-path heuristic since it had easily-defined subsets that had very little overlap and linkages between subsets. Furthermore, artificial changes to the domain decreased the number of links between subsets: instances that could have been in multiple subsets were often redefined separately in each, and the access permissions on the different areas made linking across subsets difficult. Though we feel that though these circumstances have probably inflated the results somewhat, this algorithm would still perform reasonably without the artificial changes. However, not all domains are likely to have such neatly separated relevant subsets, and the performance of this simple algorithm on such a domain is unknown.

Discussion

While our experiment does not show that the simple shortest-path algorithm presented would be adequate in general, it does show that there is promise in exploring relatively ontology-neutral methods for deriving relevant subsets. Indeed, for the investigations modeled in SemanticOrganizer, we could have used this method to derive the subset of instances relevant to each investigation with excellent results. Though we have not extended the algorithm to consider individual knowledge statements instead of instances, we could do so by including all knowledge statements that refer only to instances in the relevant subset and excluding all that refer to instances outside the subset.

Ultimately, we do not believe that ontology-neutral automated techniques alone will be adequate in most cases. Rather, we suggest that they could be used to generate an initial, rough cut of the relevant subset that could then be refined. For instance, the relevant subset could be further refined by using additional user defined descriptions to add or subtract from the relevant subset. Presumably, such "corrective" abstractions would be simpler to engineer than those that start from scratch. One interesting possibility would be to use the automatically derived subsets to generate the initial abstraction as a starting point, i.e., generating a description that defines a subset that closely matches the automatically derived subset. Finally, if additional refinements were needed, the subsets could be adjusted manually- again with considerably less overall effort than if the entire effort had been manual.

Future work

We have explored the use of a general workspace derivation technique that is independent of a given ontology, but much work remains to develop widely applicable techniques. One possibility for follow-on work would be to continue to evaluate the simple shortest-path algorithm in other domains, and to more fully evaluate its performance in the given experiment. The shortest path algorithm could readily be expanded to a weighted path algorithm that gives different weights for different links, perhaps based on the ontology or other characteristics. Furthermore, the current algorithm should be extended to apply to individual knowledge statements instead of instances and then evaluated.

Other techniques for deriving the relevant subset should also be explored. Heuristics that are not based on properties of the graph but on information retrieval methods, such as TF-IDF, are a possibility. In addition, standard machine learning methods could be explored, such as traditional clustering techniques adapted to a Semantic Web framework or relational data mining methods. We have restricted our experiments to deriving relevant subsets defined by domain, but other kinds of relevant subsets should be considered, for instance subsets defined by a specific task, goal, or timeframe. Finally, incorporating some amount of semantic interpretation into these approaches, as well as having them interact with manually derived abstractions, are directions that we feel will ultimately be the most successful.

Conclusion

As the Semantic Web gains in popularity and acceptance, it will also grow in size. To date, few semantic repositories have grown to a size that their usability suffers, but SemanticOrganizer is one such example. For the vision of the Semantic Web to be realized, these issues of scale must be addressed. We have presented one definition of a restricted view on a semantic network, which we have called a *workspace*. In essence, a workspace is the intersection of three sets; what you have permission to see,

what you can understand, and what is relevant in the current situation. Of these three concepts, we felt the latter, what is *relevant*, was the one most in need of our attention in the context of the developing Semantic Web. We have described some of the techniques that can be used to derive these relevant subsets, and have shown that even a very simple approach with minimal semantic interpretation can be successful in some domains. Ultimately, though, we feel that effective methods will require a combination of both domain independent and domain specific approaches.

Acknowledgements

We would like to thank Ian Sturken, Dan Berrios, and the SemanticOrganizer team for their contributions to this paper. Our work on SemanticOrganizer was funded by the NASA Intelligent Systems Project within the Computing, Information and Communications Technology Program and by the Investigative Methods and Tools Project within the Engineering for Complex Systems Program.

References

Berners-Lee, T., Hendler, J., and Lassila, O. (2001). The Semantic Web. *Scientific American*.

Carvalho, R. E., Williams, J., Sturken, I., Keller, R. M., and Panontin, T. Investigationorganizer: The Development and Testing of a Web-Based Tool to Support Mishap Investigations. In *Proceedings of the IEEE Aerospace Conference 2005*, Big Sky, MT, USA.

Haase, P., Broekstra, J., Eberhart, A., and Volz, R. A Comparison of Rdf Query Languages. In *Proceedings of the Third International Semantic Web Conference (ISWC-2004)*, Hiroshima, Japan.

Kalfoglou, Y., and Schorlemmer, M. (2003). Ontology Mapping: The State of the Art. *The Knowledge Engineering Review*, 18(1), 1-31.

Keller, R. M., Berrios, D. C., Carvalho, R. E., Hall, D. R., Rich, S. J., I. B. Sturken, Swanson, K. J., and Wolfe, S. R. Semanticorganizer: A Customizable Semantic Repository for Distributed Nasa Project Teams. In *Proceedings of the Third International Semantic Web Conference (ISWC-2004)*, Hiroshima, Japan.

Maganaraki, A., Tannen, V., Christophides, V., and Plexousakis, D. (2004). Viewing the Semantic Web through Rvl Lenses. *Journal of Web Semantics*.

Noy, N. F. (2004). Semantic Integration: a Survey of Ontology-Based Approaches. *ACM SIGMOD Record*, 33(4).

Noy, N. F., and Musen, M. A. Specifying Ontology Views by Traversal. In *Proceedings of the Third International Semantic Web Conference (ISWC-2004)*, Hiroshima, Japan.

Salton, G. (1983). *Introduction to Modern Information Retrieval*, McGraw-Hill.

Van Rijsbergen, C. J. (1979). *Information Retrieval (2nd Edition)*, Butterworths, London.

Volz, R., Oberle, D., and Studer, R. On Views in the Semantic Web. In *Proceedings of the Proceedings of the 2nd International Workshop on Databases, Documents, and Information Fusion (DBFUSION 02)*, Karlsruhe, Germany.

An automatic ontology-based approach to enrich tables semantically

Hélène Gagliardi, Ollivier Haemmerlé, Nathalie Pernelle, Fatiha Saïs
LRI (UMR CNRS 8623 - University of Paris-Sud),
Bâtiment 490, F-91405 Orsay Cedex, France
{gag,pernelle,sais}@lri.fr, Ollivier.Haemmerle@inapg.inra.fr

Abstract

This work aims at building automatically a thematic data warehouse composed of heterogeneous XML documents extracted from the Web. We focus on the data tables contained in these documents. This article presents how we enrich semantically those tables by means of tags and values coming from the ontology of the application. First results are given for a set of real data of the e.dot project.

Introduction

Our work deals with the automatic construction of domain specific data warehouses. More precisely, our goal is to integrate automatically information found on the Web with existing information stored in different databases. The first originality of our work is that the unique external source of knowledge used to extract information is an ontology of the application domain. Then our approach is completely generic. The second originality is that the extraction of information is done in a completely automatic manner. The drawback of such a non-supervised approach is that it leads to ambiguities or misunderstandings in the information we discover. But we propose to keep the different interpretations in order to allow their use during the query processing. That flexibility is the third originality of our technique. The fourth originality is that we exclusively extract information from data tables in the documents we found on the Web. Such a choice can appear as restrictive, but in a large variety of scientific fields, we saw that data tables contain synthetic and reliable information. Finally, our approach is currently under test in a real and ambitious project concerning the microbiological risk in food products.

Our application domain concerns the microbiological risk in food products. In order to understand and to prevent such risks, the Sym'Previus project has been launched by French governmental institutions. During the Sym'Previus project, the MIEL++ system has been built (Buche *et al.* 2004). MIEL++ is a tool based on a database, containing experimental results and industrial results about the behaviour of pathogenic germs in food products depending on several parameters, such as the temperature, the pH, etc. The Sym'Previus database is incomplete by nature since the number of possible experiments is potentially infinite. The work presented in this article takes place within the e.dot project, which is a cooperation between the INA P-G/INRA MIA group, the Xyleme start-up, the IASI-Gemo team (LRI) and the Verso-Gemo team (INRIA-Futurs). The goal of the e.dot project is to palliate the incompleteness of the database by complementing it with data automatically extracted from the Web. The drawback of such a technique is that the way the data are expressed on the Web is very heterogeneous. For example, the terms used in the scientific articles in microbiology can be different from an article to another. A way of solving that heterogeneity issue can be to query the existing database and the Web documents through a mediated architecture based on a domain ontology.

In MIEL++, the database is queried through a mediated architecture (2 local bases previously developed during the Sym'Previus project, and expressed in heterogeneous formalisms are actually queried on). The mediated schema is composed of an ontology called the Sym'Previus ontology. In order to make possible the query processing on the data extracted from the Web, we need to translate these data in order to make them compatible with the Sym'Previus ontology used in the mediated schema. That mechanism is presented in this article.

In e.dot project (e.dot 2004), data are acquired by going through the following steps. First, a Web crawler is combined with a filtering tool (Mezaour 2005) that selects the Web pages that contain data useful for the warehouse. We exclusively focus on documents in Html or Pdf format which contain data tables; actually data tables are very common presentation scheme for authors in order to describe experimental results, statistical or other synthetic data in scientific articles. In our system, these tables are extracted and transformed in a generic XML representation called XTab. These documents are then semantically enriched and stored in the data warehouse.

In this paper, we present the semantic enrichment step. In our approach, we want this transformation to be as automatic and flexible as possible, only driven by the ontology and the way the data have been structured in the original table. Thus, we have defined a Document Type Definition named SML (Semantic Markup Language) which can automatically be generated using the ontology and which can

deal with additional or incomplete information in a semantic relation, ambiguities or possible interpretation errors. This approach has been implemented and tested on real data from the e.dot project.

The paper is structured as follows. In section 2, we first introduce the XTab format, the Sym'Previus ontology, and a simple example in order to explain the aims of the semantic enrichment task. Section 3 introduces the way we identify the ontology terms represented by the columns of a table. Section 4 presents the identification of semantic relations in data table, while section 5 explains the instantiation of such semantic relation. Section 6 gives an idea of the possible use of the semantic enrichment during the query processing. In section 7, some experimental results are shown. In the conclusion, we present related works and we give future directions.

Preliminary notions

We first present the generic XML representation of tables – called XTab. Then we introduce the ontology of the application domain. That section ends with a very preliminary example of what the result of the semantic enrichment is.

The XTab format

The data tables are first represented in XML, using purely syntactic tags that are domain-independent. The tables are automatically represented using a list of lines, each line being composed of a list of cells. Besides, when it is possible, titles are extracted. This format called XTab has been defined in the e.dot project (e.dot 2004). More complex structures of tables need heuristics such as (Pivk, Cimiano, & Sure 2004) in order to be translated into this simple XTab structure. These heuristics are not presented here. The XTab representation of Figure 1 is shown in Figure 2.

Products	pH values
Cultivated mushroom	5.00
Crab	6.60

Figure 1: *approximative pH of some food products*

The Sym'Previus ontology

The Sym'Previus project (sym) has developed an ontology dedicated to the risk assessment domain. In order to exploit the data tables and query them through the MIEL++ system – which is based on the Sym'Previus ontology – we have to express data using the vocabulary stored in that ontology. The Sym'Previus ontology is composed of:

1. a term taxonomy which contains 428 terms of the domain (food, microorganism, experimental factors, ...) which are organized by the specialization relation $\preceq$;
2. a relational schema that contains 25 semantic relations between terms of the taxonomy. A semantic relation r is characterized by its signature $attrs(r)$ composed of the set of attributes of the relation. The elements of $attrs(r)$ belong to the term taxonomy. For instance, the relation *foodFactorMicroorganism* has the signature *(food, factor, microorganism).*

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<table><title> <table-title>
approximative pH of some food products</table-title>
<column-title>Products</column-title>
<column-title>pH values</column-title></title>
<nb-col>2</nb-col>
<content>
<line>
<cell>cultivated mushroom</cell>
<cell>5.00</cell>
</line>
<line>
<cell>crab</cell>
<cell>6.60</cell>
</line>
</content></table>
```

Figure 2: *XTab Representation of figure 1*

Very preliminary example

Thus, we enrich XTab documents with tags and values provided by the ontology. More precisely, we have defined a representation formalism named SML – *Semantic Markup Language* – where table lines are not represented by cells anymore but by a set of semantic relations between columns.

```
<table> <title><table-title>
approximative pH of some food products </table-title>
<column-title> Products </column-title>
<column-title>pH values</column-title>...
</title> <content>
<rowRel>
<foodPH>
<food><ontoVal>mushroom</ontoVal>
<originalVal> cultivated mushroom </originalVal>
</food>
<ph><ontoVal/>
<originalVal>5.00</originalVal></ph> </foodPH>
</rowRel>
<rowRel>
<foodPH>
<food><ontoVal>crab</ontoVal>
<originalVal>crab</originalVal></food>
<ph> <ontoVal/><originalVal>6.60</originalVal> </ph>
</foodPH> </rowRel>
</content> </table>
```

Figure 3: *Simplified SML Representation of Figure 1*

Let us consider the semantic relation named *foodPH* which links a food product with its pH value in the ontology. The aim of the enrichment is to reformulate an XTab document such as Figure 2 in an SML document such as Figure 3. In this SML document, the semantic relation *foodPH* which has been recognized in the table is represented and instantiated using the table values.

In order to instantiate the relation, we try to associate one

or several terms of the taxonomy with each value of the table. If the value does not appear directly in the taxonomy, we use mapping techniques in order to find similar terms. In the example, the first column value *crab* belongs to the taxonomy. But the value *cultivated mushroom* does not appear in the taxonomy; nevertheless, we propose to associate *mushroom* with it thanks to a mapping procedure. This value is represented in the SML tag $< ontoVal >$ while the original value is kept using the tag $< originalVal >$. Thus, the original value can be shown in the result of a query, even if the query is asked on a value belonging to the ontology. This SML representation conforms to the SML DTD (Document Type Definition) we have defined in (e.dot 2004).

Identification of the columns of the data table

In order to extract the relations of the table, we perform two steps. The first one, presented in this section, consists in identifying a term of the taxonomy which represents each column of the data table. The second step, presented in the next section, will consist in discovering semantic relations between data table organized in columns.

The identification of the columns of the data table is based on two pieces of information: the content of the column which is mainly used, and the title of the column, which is used in case the content of the column is not helpful enough.

The content of the column is used as follows: we try to associate a term of the ontology taxonomy with each value belonging to the column. Then we search for common generalizers – "subsumers" of these terms. The use of a threshold allows us to associate a generalizer with a given column even if we have not recognized all the values of that column.

definition An *A-term* is a term of the taxonomy that appears at least one time as an attribute of a relation signature in the relational schema of the ontology. The set of all A-terms is noted *AT*.

We first try to find values of the columns that belong to the taxonomy or that are included[1] in one term of the taxonomy. We then look for an A-term which subsumes almost all the values in the term taxonomy. First, an A-term can be associated with a column Col if and only if the rate of the subsumed values is greater than a given threshold th. The set of all A-terms that verify this constraint is noted $ATCandidate(Col, th)$:

$$ATCandidate(Col, th) = \{t \,|t \; in \; AT \; and \; \frac{|sub(t, Col)|}{|Col|} \geq th\}$$

where ***sub(t,Col)*** is the set of values of Col that are subsumed by the A-term t.

Among these candidates, we select the most specific A-terms that subsume the largest set of values. This set of representative A-terms is noted *ATRep*:

$$ATRep(Col, th) = \{t \mid t \in ATCandidate(Col, th),$$
$$\neg\exists\, t' \; such \; that \; t' \in ATCandidate(Col, th)$$
$$and \; |sub(t', Col)| > |sub(t, Col)|,$$
$$\neg\exists\, t'' \; such \; that \; t'' \in ATCandidate(Col, th)$$
$$and \; |sub(t'', Col)| = |sub(t, Col)| \; and \; t'' \preceq t\}$$

If there is more than one A-Term in ATRep, we keep the first one. In fact, experiments have shown that if the threshold is high enough there is zero or one representative A-term.

If no representative A-term has been found by using this procedure, we exploit the title of the column if it is available. We exploit the values of the column first because if we are able to identify an important number of values, the A-term is often relevant. Besides, the treatment of the title can lead us to a misunderstanding association. If no A-Term has been found, we keep the column in the SML document and we associate the generic A-term named *attribute* with it.

Products	Qty	Lipids	Calories
whiting with lemon	100 g	7.8 g	92 kcal
ground crab	150 g	11.25 g	192 kcal
chicken	250 g	18.75 g	312 kcal

Figure 4: *Nutritional Composition of some food products*

In the table of Figure 4, the terms *crab* and *chicken* belonging to the ontology have been associated with the values *ground crab* and *chicken*. If the threshold is 0.5, the most specific A-term that subsumes these two terms is the A-term *Food*. The second column has not been identified because it only contains numeric values and the title is an abbreviation; the generic A-Term *attribute* is associated with it. *lipid* and *calorie* have been associated with the last two columns thanks to the exploitation of their titles.

Definition The schema *tabSch* of a table *tab*, noted *tabSch(tab)*, is the finite set of couples $(col, ATRep(col, th))$ that can be found for a given threshold th.

$$tabSch(Tab) = \{(col, t) | t \in ATRep(col, th)$$
$$or \; [(t = attribute) \; and \; ATRep(col, th) = \emptyset]\}$$

The schema of the table *Tab2* shown in Figure 4 is:
tabSch(Tab2)={(1,food) (2,attribute) (3,lipid),(4,calorie)}

Identification of the semantic relations appearing in the data table

We present now how we identify one or several semantic relations in the schema of the table. That identification is done by comparing the "natures" of the columns identified during the previous step with the attributes appearing in the signatures of the semantic relations of the ontology of the domain. Of course, an exact mapping between the schema of the table and the signature of a specific semantic relation is the ideal case. In most of the cases, we will obtain several possible mapping with subsets of the

[1] in the sense of the inclusion of sets of words, after a lemmatization step and without taking the "empty" words (determiners or prepositions) into account

attributes of the schema of the table. Or we will have only partial mapping, with only a subset of the attributes of the signature of a relation, etc. So we will see that we propose an automatic identification of the semantic relations as flexible as possible.

We say that a relation is ***completely represented*** if each attribute of its signature subsumes or is equal to a distinct A-term of the table schema.

Thus, suppose that the three relations *foodLipid*, *foodCalorie*, *foodPh* belong to the ontology and that the two relations *foodLipid* and *foodCalorie* mean "the number of lipid (or calories) contained in 100 g of the foodstuff", because the experts have considered that the weight is normalized. In table of Figure 4, the relations are extracted in the following way:

- *foodLipid*, is completely represented by the values found in the first and the third columns.
- *foodCalorie* is completely represented by the values found in the first and the fourth columns.

Since the second column *qty* is not identified and does not participate to any of these two relations, we add to each relation a generic attribute which will contain values found in this second column. If this attribute was not represented, for example, the third line of the table would be interpreted as "***100g*** *of chicken correspond to 312 calories*". When the generic attribute is taken into account, the interpretation is "***250g*** *of chicken correspond to 312 calories*". So, in such cases, the representation of additional information leads to better interpretations of the data.

Figure 5 proposes the SML representation of the relations *foodLipid* and *foodCalorie* :

```
<table> <content>
<rowRel additionalAttr="yes">
<foodLipid relType ="completeRel">
<food>...</food> <lipid> ... </lipid>
<attribute> ...</attribute>
</foodLipid>
<foodCalorie relType ="completeRel">
<food>...</food> <calorie> ... </calorie>
<attribute> ...</attribute>
</foodCalorie>
</rowRel> ...
</content> </table>
```

Figure 5: *SML representation of completely represented relations*

We say that a relation is ***partially represented*** if it is not completely represented and if at least two attributes of its signature subsume or are equal to different A-terms of the schema of the table. We have considered partially represented relations in order to take the following two cases into account.

Partially represented relations with Null attributes:

This is the case when an attribute of the semantic relation has not been associated to column of the table schema. For example in the table of Figure 4, the semantic relation *foodAmountLipid*, defined in the ontology on its attributes *food*, *amount* and *lipid*, is partially represented in the table schema *tabSch*, since the attribute *amount* is not represented in the table schema. Figure 6 presents the SML representation of *foodAmountLipid* relation :

```
<table> <content>
<rowRel additionalAttr="yes">
...
<foodAmountLipid relType ="partialNull">
<food attrType="Normal">...</food>
<amount attrType="Null"/ >
<lipid attrType ="Normal"> ... </lipid>
<attribute attrType="generic"> ...</attribute>
</foodLipid>
</rowRel> ... </content> </table>
```

Figure 6: *SML representation of a partially represented relation with Null attributes*

Note that when a relation is partially represented, the attributes that do not appear in the schema are represented in the SML document by means of an empty tag like <*amount* ***attrType="Null"***/>. In this example, the generic attribute represents precisely the missing attribute *Amount*.

Partially represented relations with constant values:

This is the case when one of the relation attributes correspond to a constant value which appears in the title of the table.

Products	Doubling time (h)
Minced meat	30[1]
Cured raw pork	3.6[1]
Frankfurters	9[1]

Figure 7: *Doubling times of Listeria monocytogenes in foodstuffs*

Let *tabSch* the table schema computed from the table tab_3 of Figure 7: $tabSch(tab_3) = \{(1,food),(2,factor)\}$.

In this table schema, the relation *foodFactorMicroorganism* is partially represented: the attributes *food* and *factor* are represented in the table schema and the attribute *Microorganism* is represented by a constant value *Listeria Monocytogenes* which appears in the table title "Doubling time of ***Listeria Monocytogenes*** in foodstuffs".

This constant is used as a value for the corresponding attribute of the semantic relation and it is propagated into all the instances of the relation. Figure 8 presents the SML representation of the *foodFactorMicroorganism* relation.

Because we want to keep unidentified data, we also add to the semantic relations we have found the set of generic attributes of the table schema. This is done even if the relation is partial. Actually, one of these additional attributes may be

```
<table>
<content>
<rowRel additionalAttr="no">
...
<foodFactorMicroorganism relType ="partialConst">
<food attrType="Normal">...</food>
<factor attrType="Normal"> ... </factor>
<microorganism attrType ="Const"> listeria monocytogenes
</microorganism>
</foodFactorMicroorganism>
...
</rowRel> ...
</content> </table>
```

Figure 8: *SML representation of partially represented relations with attributes in constants*

a missing attribute of the relation. Besides, this attribute can add a contextual information which may modify the user's interpretation of the relation.

When no relation has been found in the table schema, a generic relation named *relation* is generated in the SML document. In this way, we keep semantic links between values even if this link has not been identified. Thus, it is possible to query the SML documents by means of lists of key-words.

Instantiation of the semantic relations

Once the relations are extracted, we instantiate them by the values contained in the table. Besides, terms of the ontology are associated with each value when it is possible. The SML formalism allows us to associate several terms that can be found by different mapping mechanisms. We have considered two kinds of mapping procedures.

The first one uses simple syntactic criteria. Each value is considered as a set of lemmatized words M_v where empty words such as determiners or prepositions are suppressed. The same treatment is applied to the terms of the ontology. Then, we consider that there may exist a semantic similarity between a value v and a term t if :

1. equality: ($M_v = M_t$)
2. inclusion: ($M_v \subset M_t$ or $M_t \subset M_v$)
3. intersection: ($M_t \subset M_v$) or ($M_v \cap M_t \neq \emptyset$).

These three criteria are applied using the previous order.

The second mapping procedure uses more semantic criteria. Actually, we have chosen to use the unsupervised approach PANKOW – Pattern-based Annotation trough Knowledge On the Web (Cimiano, Handschuh, & Staab 2004) where patterns are used to categorize proper nouns (instances) with regard to an ontology. PANKOW applies a set of linguistic patterns including Hearst patterns (Hearst 1992) (i.e. the $< concept >< instance >$, $< concept >$ such as $< instance >$, ...) on the biggest corpus available: the World Wide Web. In fact, they exploit the google API and take the number of pages in which patterns appear as an indicator for the strength of the pattern. We have used the same approach on data table even if they are not necessarily proper nouns. We have applied the general pattern "$< value >$ is a $< term >$" in order to discover specialization relations between values and terms of the ontology using the Web corpus. For a given value, we instantiate the pattern with each term of the domain ontology and keep the best term with regard to the number of pages. Because of the specificity of our domain, the number of pages can be very low. For instance, when we try to associate the value "ice cream" to a term of the ontology, the pattern "ice cream is a dessert" is found in 35 pages. Happily, "ice cream is a microorganism" is not found. Note that the term *dessert* cannot be found by our syntactic criteria.

Figure 9 shows a part of the SML document which is automatically generated from the XTab document of Figure 4. This document is structured in the following way:

```
<table> <table-title>Nutritional Composition of some food products </table-
title >
<column-title> Product </column-title> <column-title>Qty</column-title>
<column-title>lipids</column-title>
<column-title>calories</column-title> <column-nb> 4 </column-nb>
<content>
<rowRel additionalAttr="yes">
<foodLipid relType="completeRel">
<food indProc="yes" attrType="Normal">
<ontoVal indMap="intersection"> whiting Provencale
</ontoVal>
<ontoVal indMap="intersection"> green lemon </ontoVal>
<ontoVal> whiting fillets </ontoVal>
<originalVal> whiting with lemon </originalVal>
</food>
<lipid indProc="no" attrType="Normal">
<ontoVal indMap="notFound"/>
<originalVal> 7.8 g</originalVal>
</lipid>
<attribute indMap="notFound" indProc="no"
attrType="Generic"> <ontoVal/>
<originalVal> 100 g</originalVal></attribute>
</foodLipid>
<foodCalorie relType="completeRel"> ... </foodCalorie>
<foodAmountLipid relType="partialNull"> ...
</foodAmountLipid> </rowRel> ... </content> </table>
```

Figure 9: *SML Representation of the nutritional composition of food products*

The main part of the document is inside the ***content*** element. It represents the table like a set of lines where each line is now a set of semantic relations (like, for example, ***foodLipide or foodCalories***).
The SML representation of a relation is composed of the set of attributes that appear in the signature of the relation described in the relational Reference Schema of the ontology (e.g. *foodLipid*(*food*, *lipid*)). Each attribute subsumes the representative term of the column or subsumes a term which has been found in its title. A set of terms represented inside the XML tag *ontoVal* is associated with each value. Thus, *crab* has been associated with *ground crab* while three different terms are proposed for *whiting with lemon* : *whiting*

Provencale, *green lemon* and *whiting fillets*. The original value is kept inside the XML tag *originalVal*.

The generality of the SML representation is ensured by the possibility of an automatic generation of the SML DTD from an ontology which contains a taxonomy and a relational reference schema. In the following, we give an example of relational schema and its corresponding SML DTD.

In the figure 10 we present an extract of the relational schema of an ontology of the risk assessment. Figure 11 is the corresponding representation of the SML DTD generated from this relational schema. The DTD is simply represented here as a graph.

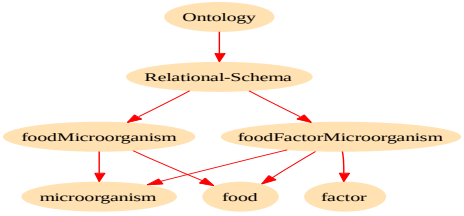


Figure 10: Extract of a risk assessment ontology

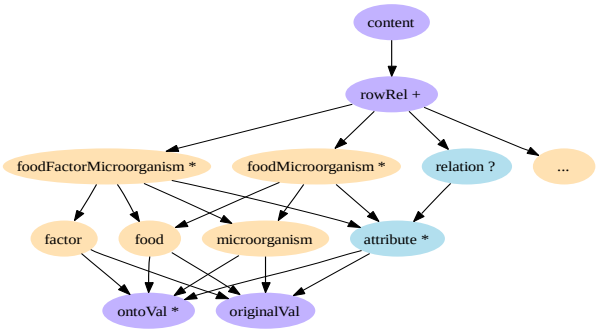


Figure 11: Extract of SML DTD (risk assessment)

Interrogation of SML documents

Some indicators that can be exploited in the queries

Our approach allows one to extract data from tables even if we are not sure of their representation using the vocabulary of the ontology. It is the reason why we have defined a list of indicators that are represented in the SML document and that will be exploited during the query evaluation.

We present now the two main treatment indicators represented in SML as XML attributes attached to lines or to relation attributes. The first one is related to the structure of the relations (presence or absence of additional attributes).

additionalAttr: it informs on the presence of one or several additional attributes that represent the columns of the table which could not be associated with an identified relation. It is added to the tags $< rowRel >$ of SML document. For example in the table of Figure 4, this indicator allows the query engine to use the generic attribute associated with the *Quantity* column.

The following indicator make it possible to specify the kind of mapping procedure used to find a term of the ontology; it can thus be used to evaluate the risk of a mapping error. It is added to the $< ontoVal >$ tags of the SML document.

indMap: it indicates the name of the mapping procedure (inclusion, intersection or PANKOW) used to find the term of the ontology which corresponds to the original value of the table. Several mapping operators can exist in the application, this indicator allows us to modulate a trust degree, relating to enrichment, according to mapping operators. Besides, it can be used to visualize the original value if necessary.

These treatment indicators can be used by the query engine to adapt and find other answers for the user in cases of dissatisfaction.

An example of interrogation

To query SML documents, XQuery queries have been written. They rely on the SML DTD. In the following, we describe a query example where the user looks for the quantity of lipid in 100 g of crab. The evaluation of this query consists in searching in the SML document for the subtrees – SML fragments – such that the parent node is *foodLipid* and such that there is an element *ontoVal* that contains the value “crab”. The indicators *indMap* and *indProc* are used to check the validity of the semantic enrichment of the data. As the indicator *additionalAttr* has the value “yes”, the query engine displays the additional information *150g*. This example shows how the unidentified attributes that are kept in the SML representation can increase the accuracy of the user interpretation. Besides, the original value *ground crab* is displayed since *indProc* indicates that a treatment was carried out on the original value. The evaluation of this query performed on the document of Figure 9 is presented in Figure 12.

<table>
<title> Nutritional composition of some food products </title>
<food> **ground crab**</food> <lipid>**11.25 g**</lipid>
<validity>inclusion</validity>
<additionalattr>**150 g**</additionalattr>
<category> unknown</category> </table>

Figure 12: *A possible structure of the query answer*

First results

We present in this section the results of the first experimentation of our method. The approach has only be tested on the risk assessment domain represented in the Sym’Previus ontology. In this evaluation, we show the capacity of our system to recognize relations of the ontology in the XTab

tables . Our goal was to compare the results provided with our automatic method with a manual one done by an expert. We compared the results in terms of the well-known information retrieval measures Precision, Recall and F-Measure.

Test set

Among two hundred real XTab tables collected from the Web, we have selected 33 tables. One table is selected in the test set if and only if we identify, among its columns at least one semantic relation attribute represented in the ontology.

Evaluation methodology

In order to evaluate our approach, we have distinguished the results found for the three kinds of semantic relations: the Completely represented Relations (*CR*), the Partially represented Relations where all the missing attributes are identified by Constants in the table title (*PRC*) and the Partially represented Relations which contain at least one attribute which is not identified – Null attributes – (*PRN*). Note that PRC relation can only found in the tables which are associated with a table title.

In first step we run our prototype on the real test set of XTab documents. In second step a domain expert checks the relevance of each semantic relation provided by our system.

To identify the semantic relations represented in the table represented in the XTab document, the expert has access to the whole information of the original – *HTML or Pdf* – document but he only considers information which are contained in the XTab document (ie. the table title and the table content). The expert considers that a semantic relation is *correct* if the relation is represented in the table and if all its attributes are correctly identified. If he recognizes in the XTab document one semantic relation which is not found by our system, he considers that the relation is *forgotten*. By this way, he can determine which semantic relations provided by our system are incorrect and which are forgotten.

In Figure 13, we show the result of this step for each kind of relation (CR, PRC and PRN) : number of semantic relations which have been found by the system, incorrect semantic relations and forgotten semantic relations.

	Found rels	**Incorrect rels**	**Forgotten rels**
CR	30	22	11
PRC	6	3	5
PRN	23	2	3

Figure 13: *Expert results after semantic relations checking step*

On these results we have computed Recall, Precision and F-Measure. Let *T, T'* be two variables that represent the semantic relation type considered in the three measures calculations. It gets values in : {CR, (CR and PRN), (CR, PRN and PRC)}. Let *Correct_Rels(*T*)* be the number of semantic relations of type *T*, correctly found by our system.

[1]The XTab tables are the result of an automatic transformation applied on HTML and PDF documents found on the Web

$$Correct\_Rels(T) = Found\_Rels(T) - Incorrect\_Rels(T)$$

Recall is the percentage of relations (all types) actually represented in the data tables and correctly found by our system. Here we suppose that *T'*={*CR, PRN and PRC*}.

$$Recall = \frac{Correct\_Rels(T)}{(Correct\_Rels(T')) + Forgotten\_Rels(T'))}$$

Precision Is the percentage of relations found in the data tables by our system and with a correctly assigned relation signature.

$$Precision = \frac{Correct\_Rels(T)}{Found\_Rels(T)}$$

F-Measure as usual we balance Recall and Precision against each other.

$$F - Measure = \frac{2 * Recall * Precision}{Recall + Precision}$$

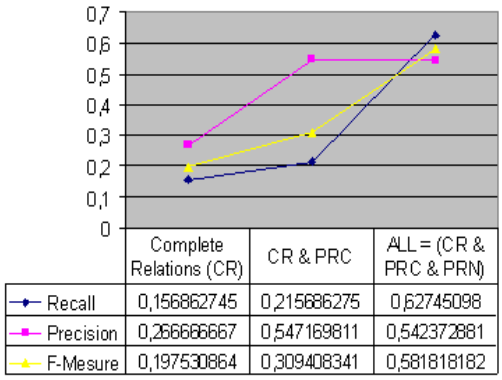


Figure 14: Recall, Precision and F-Measure for a threshold at 0.3

Results

The diagram presented in Figure 14 gives the results in term of precision, recall and F-Measure of our semantic enrichment system approach. The first interesting observation is that the recall value increases significantly when our system takes into account partially represented relations. This result shows clearly the interest of the partially identified semantic relations kept in the SML documents, even when missing attributes are not identified by constants. If we restrict the relation types on the complete relations, we would have only 0.15 for the recall value, whereas in the case where we keep all the identified relations (ie. completely and partially represented relations) we have 0.62 for the recall value. Note that our aim is precisely to obtain a satisfying *Recall* value. Because we have chosen to keep all the identified pieces of information as well as information which are not completely identified such as partial relations, generic attributes end partial relations. We can also note that the precision is increasing as well. This result is globally shown by the increasing of the F-Measure value.

Conclusion

Our method allows one to enrich semantically documents found on the Web which present the specificity of a tabular structuring. The semantic enrichment is completely automatic and it is guided by an ontology of the domain. Thus, that processing cannot lead to a perfect and complete enrichment. The XML representation we propose keeps all the possible interpretation in order to let the possibility of using them during the query step, for example by allowing a query processing based on keywords or by exhibiting some relevant information to the user in order to help him/her during the interpretation of the results.

Then, in case of ambiguity, it is possible to associate several terms of the ontology or several semantic relations with a same set of columns. In order to allow the query processor to adapt its answers or to evaluate their relevance, we log the processes by means of a set of indicators. The generality of our approach is ensured by the fact that the SML DTD can be automatically generated from the ontology.

The approach we propose is currently under testing in the domain of the food risk assessment, by means of a Java prototype. In order to query SML documents, we wrote *XQuery* queries which take advantage of the treatment indicators inserted in the SML documents. Those queries have been tested by means of the MIEL++ query engine.

Some works like (Kushmerick 2000), (Muslea, Minton, & Knoblock 2001) and (Hsu & Dung 1998) allow to extract knowledge by learning rules from a sample of manually annotated documents. Our goal is quite different since our approach is completely automatic and exclusively guided by the ontology.

Moreover, the documents we use to fill the data warehouse are heterogeneous and, contrarily to previous approaches like (Crescenzi, Mecca, & Merialdo 2002) and (Arasu & Garcia-Molina 2003), we cannot base the search for information on a common structure discovered among a set of homogeneous documents.

The techniques we use to identify the columns of the table are based first on the values contained in those columns. (Rahm & Bernstein 2001) and (Doan *et al.* 2003) showed that those techniques give good results in the framework of the search for schema mappings for relational databases or XML. In our case, we do not have the schema of the tables we work on: we have to discover it first before searching for mappings with the semantic relations of the ontology.

We can now enhance our mapping operators, for example by using external resources such as WordNet or by using more sophisticated similarity measures (Robertson & Willett 1998). Moreover, we can think about using linguistic tools allowing to process the table content (cells, titles) represented in a more complex way. We also want to check the generality of our approach by applying it to another application domain.

References

Arasu, A., and Garcia-Molina, H. 2003. Extracting structured data from web pages. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, 337–348. ACM Press.

Buche, P.; Dibie-Barthélemy, J.; Haemmerlé, O.; and Houhou, M. 2004. Towards flexible querying of xml imprecise data in a dataware house opened on the web. In *Flexible Query Answering Systems (FQAS)*. Springer Verlag.

Cimiano, P.; Handschuh, S.; and Staab, S. 2004. Towards the self-annotating web. In *WWW '04: Proceedings of the 13th international conference on World Wide Web*, 462–471. ACM Press.

Crescenzi, V.; Mecca, G.; and Merialdo, P. 2002. Automatic web information extraction in the roadrunner system. In *Revised Papers from the HUMACS, DASWIS, ECOMO, and DAMA on ER 2001 Workshops*, 264–277. Springer-Verlag.

Doan, A.; Lu, Y.; Lee, Y.; and Han, J. 2003. Profile-based object matching for information integration. *Intelligent Systems, IEEE* 18(5):54– 59.

e.dot. 2004. Progress report of the e.dot project. http://www-rocq.inria.fr/gemo/edot.

Hearst, M. A. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th conference on Computational linguistics*, 539–545. Association for Computational Linguistics.

Hsu, C.-N., and Dung, M.-T. 1998. Generating finite-state transducers for semi-structured data extraction from the web. *Inf. Syst.* 23(9):521–538.

Kushmerick, N. 2000. Wrapper induction: efficiency and expressiveness. *Artif. Intell.* 118(1-2):15–68.

Mezaour, A. D. 2005. Filtering Web Documents for a Thematic Warehouse, case study : eDot a Food Risk Data Warehouse (extended). In *to Appear in Proceedings of New Trends in Intelligent Information Processing and Web Mining Conference (IIPWM'05), Gdansk, Poland*. Springer Verlag series–Advances in Soft Computing–.

Muslea, I.; Minton, S.; and Knoblock, C. A. 2001. Hierarchical wrapper induction for semistructured information sources. *Autonomous Agents and Multi-Agent Systems* 4(1-2):93–114.

Pivk, A.; Cimiano, P.; and Sure, Y. 2004. From tables to frames. In *International Semantic Web Conference*, 166–181.

Rahm, E., and Bernstein, P. A. 2001. A survey of approaches to automatic schema matching. *The VLDB Journal* 10(4):334–350.

Robertson, A., and Willett, P. 1998. Applications of n-grams in textual information systems. In *Journal of Documentation*, 48–69.

http://www.symprevius.net.

A Bayesian Methodology towards Automatic Ontology Mapping*

Zhongli Ding, Yun Peng, Rong Pan, Yang Yu

University of Maryland Baltimore County
Department of Computer Science and Electrical Engineering
1000 Hilltop Circle, Baltimore, MD 21250
{zding1, ypeng, panrong1, yangyu1}@cs.umbc.edu

Abstract

This paper presents our ongoing effort on developing a principled methodology for automatic ontology mapping based on *BayesOWL*, a probabilistic framework we developed for modeling uncertainty in semantic web. The proposed method includes four components: 1) learning probabilities (priors about concepts, conditionals between subconcepts and superconcepts, and raw semantic similarities between concepts in two different ontologies) using Naïve Bayes text classification technique, by explicitly associating a concept with a group of sample documents retrieved and selected automatically from World Wide Web (WWW); 2) representing in OWL the learned probability information concerning the entities and relations in given ontologies; 3) using the *BayesOWL* framework to automatically translate given ontologies into the Bayesian network (BN) structures and to construct the conditional probability tables (CPTs) of a BN from those learned priors or conditionals, with reasoning services within a single ontology supported by Bayesian inference; and 4) taking a set of learned initial raw similarities as input and finding new mappings between concepts from two different ontologies as an application of our formalized BN mapping theory that is based on evidential reasoning across two BNs.

Overview

Semantic heterogeneity between two different applications or agents comes from their use of conflicted or mismatched terms about concepts. Same term or concept name might have different meanings in different agents, different terms from different agents might have the same meaning, one term from an agent might matches to several or might not matches to any terms of the other agent exactly, or two terms with the same or similar meaning are structured differently in different agents (e.g., different paths from their respective root concepts). With the development of the semantic web[1], ontologies have become widely used to represent the conceptualization of a domain, i.e., concepts, properties about concepts, relations between concepts, and instances about concepts. In ontology-based semantic integration, two agents in communication need to find a way to share the semantics of the terms in their ontologies in order to fully understand each other. This can be done in several possible directions depends on the needs of particular applications: 1) one may force both agents to use a single centralized global ontology; 2) one may merge the source ontologies into one unified ontology before agent interactions; 3) one may search for a set of mappings (or matches) between two ontologies; 4) for a multi-agent system one may resolve semantic differences in runtime when they arise during agent interaction; and 5) one may translate one of the ontologies into a target ontology with the help of an intermediate shared ontology. In this context, we are particularly interested in ontology mapping. (Noy 2004) provides a brief survey about existing ontology-based approaches, which are either based on syntactic and semantic heuristics, machine learning text classification techniques by attaching a set of documents to each concept to represent its meaning, or linguistics (spelling, lexicon relations, lexical ontologies, etc.) and natural language processing techniques.

Ontology languages in the semantic web, such as OWL[2] and RDF(S)[3], are based on crisp logic and thus can not handle incomplete or partial knowledge about an application domain. However, uncertainty exists in almost every aspects of ontology engineering. For example, in domain modeling, besides knowing that "*A* is a subclass of *B*", one may also know and wishes to express that "*A* is a small subclass of *B*"; or, in the case that *A* and *B* are not logically related, one may still wishes to express that "*A* and *B* are largely overlapped with each other". In ontology reasoning, one may want to know not only if *A* is a subsumer of *B*, but also how close of *A* is to *B*; or, one may want to know the degree of similarity even if *A* and *B* are not subsumed by each other. Moreover, a description (of a class or object) one wishes to input to an ontology reasoner may be noisy and uncertain. Uncertainty becomes more prevalent in concept mapping between two ontologies where it is often the case that a concept defined in one ontology can only find partial matches to one or more concepts in another ontology.

Narrowly speaking, a mapping can be defined as a correspondence between concept *A* in Ontology 1 and concept *B* in Ontology 2 which has similar or same semantics as *A*.

* This work was supported in part by DARPA contract F30602-97-1-0215 and NSF award IIS-0326460.

1 http://www.w3.org/2001/sw/

2 http://www.w3.org/2001/sw/WebOnt/

3 http://www.w3.org/RDF/

Most existing ontology-based semantic integration approaches provide exact mappings in a semi-automatic way with manual validation, without taking the degree of uncertainty into consideration. In tackling this problem, (Mitra, Noy and Jaiswal 2004) improves existing mapping results using BNs (Pearl 1988) by a set of meta-rules that capture the structural influence and the semantics of ontology relations.

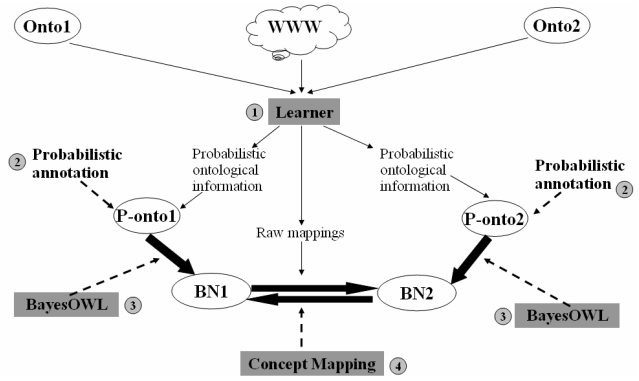


Figure 1. The System Framework

Different from their contributions, we propose a new methodology in supporting uncertainty modeling and reasoning in a single ontology, as well as ontology mapping using Bayesian networks. As can be seen from Figure 1 above, the system includes four components: 1) a learner to obtain probabilistic ontological information and raw mappings using data obtained from web; 2) a representation mechanism for the learned uncertain information concerning the entities and relations in given ontologies; 3) a *BayesOWL* (Ding, Peng, and Pan 2004; Ding and Peng 2004) module to translate given ontologies (together with the learned uncertain information) into BNs; and 4) a concept mapping module which takes a set of learned raw similarities as input and finds mappings between concepts from two different ontologies based on evidential reasoning across two BNs. The ideas about these four components, as well as their related works, are presented in the next four sections respectively. The paper ends with a discussion and suggestions for future research.

Learning Probabilities from Web Data

In this work, we use prior probability distributions *P(C)* to capture the uncertainty about concepts (i.e., how an arbitrary individual belongs to class *C*), conditional probability distributions *P(C|D)* for relations between *C* and *D* in the same ontology (e.g., how likely an arbitrary individual in class *D* is also in *D*'s subclass *C*), and joint probability distributions *P(C,D)* for semantic similarity between concepts *C* and *D* from different ontologies. In many cases these kinds of probabilistic information are not available and are difficult to obtain from domain experts. Our solution is to learn these probabilities using Naïve Bayes text classification technique (Craven et al. 2000; McCallum and Nigam 1998) by associating a concept with a group of sample documents called *exemplars*. The idea is inspired by those machine learning based semantic integration approaches such as (Doan et al. 2002; Lacher and Groh 2001; Prasad, Peng and Finin 2002) where the meaning of a concept is implicitly represented by a set of exemplars that are relevant to it.

Learning the probabilities we need from these exemplars is straightforward. First, we build a model containing statistical information about each concept's exemplars in Ontology 1 using a text classifier such as Rainbow[1], and then classify each concept in Ontology 2 by their respective exemplars using the model of Ontology 1 to obtain a set of probabilistic scores showing the similarity between concepts. Ontology 1's exemplars can be classified in the same way by model built using Ontology 2's exemplars. This cross-classification (Figure 2) process helps find a set of raw mappings between Ontology 1 and Ontology 2 by setting some threshold values. Similarly, we can obtain prior or conditional probabilities related to concepts in a single ontology through self-classification with the model for that ontology.

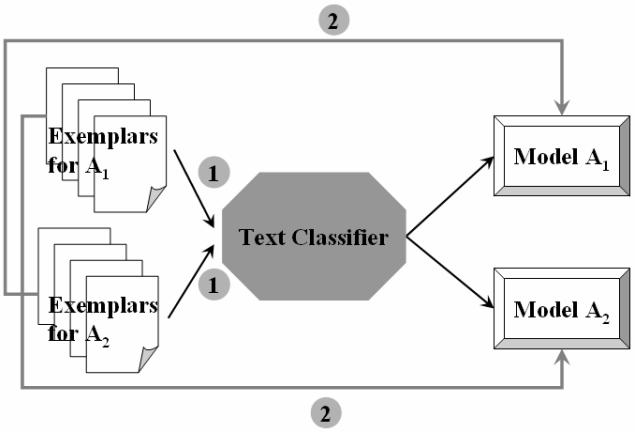


Figure 2. Cross-classification using Rainbow

The quality of these text classification based mapping algorithms is highly dependent on the quality of the exemplars (how relevant they are to the concept and how comprehensive they are in capturing all important aspects of the concept), and it would be a very time-consuming task for knowledge workers to choose high quality exemplars manually. The need to find sufficient relevant exemplars for a large quantity of concepts manually greatly reduces the attractiveness and applicability of these machine learning based approaches.

Our approach is to use search engines such as Google[2] to retrieve exemplars for each concept node automatically

[1] http://www-2.cs.cmu.edu/~mccallum/bow/rainbow
[2] http://www.google.com

from WWW, the richest information resource available nowadays. The goal is to search for documents in which the concept is used in its intended semantics. The rationale is that the meaning of a concept can be described or defined in the way it is used.

To find out what documents are relevant to a term, one can use words of the term as keywords to query the search engine. However, a word may have multiple meanings (word senses) and a query using only words of the term in attention may return irrelevant documents based on a different meaning of that word. For example, in an ontology for "food", a concept named "apple" is a subconcept of "fruit". If one only uses "apple" as the keyword for query, documents showing how to make an apple pie and documents showing how to use an iPod may both be returned. Apparently, the documents using "apple" for its meaning in computer field is irrelevant to "apple" as a fruit. Fortunately, since we are dealing with concepts in well defined ontologies, the semantics of a term is to a great extent specified by the other terms used in defining this concept in the ontology, names, the properties of that concept class, its super- and sub-concept classes. For example, if a given ontology is a concept taxonomy, the search query can be formed with all the terms on the path from root to the node in the taxonomy. By this method, the number of irrelevant documents returned is greatly reduced. In the "apple" example, the query would then become "food fruit apple" instead of "apple" itself. Documents about iPod and Apple computers will not be returned.

Search results returned by search engines are html files. There are some choices on how to use them. The simplest one is to use the entire html file as one exemplar. A second option is to use each paragraph where a keyword in the query shows up. A third option is to collect sentences containing a keyword in the html file and use this collection as an exemplar. We are currently experimenting these options, and the preliminary results suggest the second approach is the most suitable one.

Representing Probabilities in OWL

Information about the uncertainty of the classes and relations in an ontology can often be represented as probability distributions (e.g., $P(C)$ and $P(C|D)$ mentioned earlier), which we refer to as *probabilistic constraints* on the ontology. These probabilities can be either provided by domain experts or learned from web data as described in the previous section.

Although not necessary, it is beneficial to represent the probabilistic constraints as OWL statements. We have developed such a representation. At the present time, we only provide encoding of two types of probabilities: priors and pair-wise conditionals. This is because they correspond naturally to classes and relations (RDF triples) in an ontology, and are most likely to be available to ontology designers. The representation can be easily extended to constraints of other more general forms if needed.

The model-theoretic semantics of OWL treats the domain as a non-empty collection of individuals. If classe A represents a concept, we treat it as a random binary variable of two states a and $\bar{a}$, and interpret $P(A = a)$ as the prior probability or one's belief that an arbitrary individual belongs to class A, and $P(a \mid b)$ as the conditional probability that an individual of class B also belongs to class A. Similarly, we can interpret $P(\bar{a})$, $P(\bar{a} \mid b)$, $P(a \mid \bar{b})$, and $P(\bar{a} \mid \bar{b})$ with the negation interpreted as "not belonging to".

We treat a probability as a kind of resource, and define two OWL classes: "PriorProb" and "CondProb". A prior probability of a variable is defined as an instance of class "PriorProb", which has two mandatory properties: "hasVarible" (only one) and "hasProbValue" (only one). A conditional probability of a variable is defined as an instance of class "CondProb" with three mandatory properties: "hasCondition" (at least has one), "hasVariable" (only one), and "hasProbValue" (only one).

The range of "hasCondition" and "hasVariable" is a defined class named "Variable" with two mandatory properties: "hasClass" and "hasState". "hasClass" points to the concept class this probability is about and "hasState" gives the "True" (belong to) or "False" (not belong to) state of this probability.

For example, $P(c) = 0.3$, the prior probability that an arbitrary individual belongs to class C, can be expressed as

```
<Variable rdf:ID="c">
   <hasClass>C</hasClass>
   <hasState>True</hasState>
</Variable>
<PriorProb rdf:ID="P(c)">
   <hasVariable>c</hasVariable>
   <hasProbValue>0.3</hasProbValue>
</PriorProb>
```

and conditional probability $P(c \mid p1, p2) = 0.8$ can be encoded as

```
<CondProb rdf:ID="P(c|p1, p2)">
   <hasCondition>p1</hasCondition>
   <hasCondition>p2</hasCondition>
   <hasVariable>c</hasVariable>
   <hasProbValue>0.8</hasProbValue>
</CondProb>
```

with variables c, p1, and p2 properly defined.

Similar to our work, (Fukushige 2004) proposes a vocabulary for representing probabilistic relationships in a RDF graph. Three kinds of probability information can be encoded in his framework: probabilistic relations (prior), probabilistic observation (data), and probabilistic belief (posterior). And any of them can be represented using probabilistic statements which are either conditional or unconditional.

The *BayesOWL* Framework

BayesOWL (Ding, Peng and Pan 2004; Ding and Peng 2004) is a framework which augments and supplements OWL for representing and reasoning with uncertainty, based on Bayesian networks (BN). This framework provides a set of rules and procedures for direct translation of an OWL ontology into a BN structure and a method that incorporate encoded probability information when constructing the conditional probability tables (CPTs) of the BN. The translated BN, which preserves the semantics of the original ontology and is consistent with the probability information, can support ontology reasoning, both within and across ontologies as Bayesian inferences. Below we give a brief summary.

Structural Translation

A set of translation rules is developed to convert an OWL ontology (about TBox only at the present time) into a directed acyclic graph (DAG) of BN. The general principle underlying these rules is that all classes (specified as "subjects" and "objects" in RDF triples of the OWL file) are translated into nodes in BN, and an arc is drawn between two nodes in BN if the corresponding two classes are related by a "predicate" in the OWL file, with the direction from the superclass to the subclass. Control nodes are created during the translation to facilitate modeling relations among class nodes that are specified by OWL *logical* operators, and there is a converging connection from each concept nodes involved in this logical relation to its specific control node. There are five types of control nodes in total, which correspond to the five types of logical relations: "and" (owl:intersectionOf), "or" (owl:unionOf), "not" (owl:complementOf), "disjoint" (owl:disjointWith), and "same as" (owl:equivalentClass).

Constructing CPTs

The nodes in the DAG obtained from the structural translation step can be divided into two disjoint groups: X_R, nodes representing concepts in ontology, and X_C, control nodes for bridging logical relations. The CPT for a control node in X_C can be determined by the logical relation it represents so that when its state is "True", the corresponding logical relation holds among its parent nodes. When all the control nodes' states are set to "True" (denote this situation as *CT*), all the logical relations defined in the original ontology are held in the translated BN. The remaining issue is then to construct the CPTs for each node in X_R so that $P(X_R|CT)$, the joint distribution of all regular nodes in the subspace of *CT*, is consistent with all the given probabilistic constraints (which can be learned from web data as described earlier).

This is difficult for two reasons. First, the constraints are usually not given in the form of CPT. For example, CPT for variable *C* with two parents *A* and *B* is in the form of $P(C|A,B)$ but a constraint may be given as $Q(C|A)$ or even $Q(C)$. Secondly, CPTs are given in the general space of $X = X_R \cup X_C$, but constraints are for the subspace of *CT* (the dependencies changes when going from the general space to the subspace of *CT*). For example, with the constraint $Q(C|A)$, $P(C|A,B)$, the CPT for *C*, should be constructed in such a way that $P(C|A,CT) = Q(C|A)$. To overcome these difficulties, we developed an algorithm named D-IPFP (Ding, Peng, and Pan 2004) to approximate these CPTs for X_R based on the "iterative proportional fitting procedure" (IPFP), a well-known mathematical procedure that modifies a given distribution to meet a set of probabilistic constraints while minimizing *I-divergence* to the original distribution (Deming and Stephan 1940; Csiszar 1975; Bock 1989; Vomlel 1999; Cramer 2000).

Figure 3 below is a BN translated from a simple ontology. In this ontology, "Animal" is a primitive concept class; "Male", "Female", "Human" are **subclasses** of "Animal"; "Male" and "Female" are **disjoint** with each other; "Man" is the **intersection** of "Male" and "Human"; "Woman" is the **intersection** of "Female" and "Human"; "Human" is the **union** of "Man" and "Woman".

The following probability constraints are attached to X_R = {Animal, Male, Female, Human, Man, Woman}:

P(Animal) = 0.5; P(Male|Animal) = 0.5;
P(Female|Animal) = 0.48; P(Human|Animal) = 0.1;
P(Man|Human) = 0.49; P(Woman|Human) = 0.51.

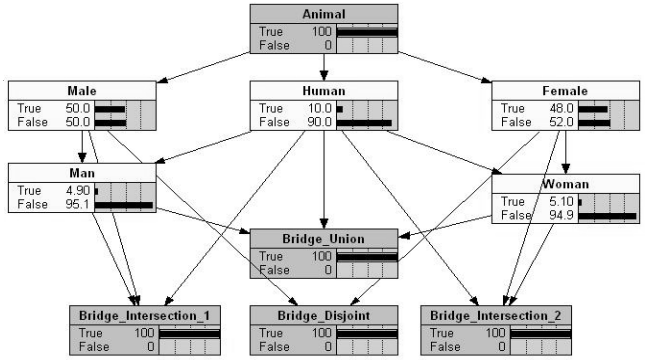


Figure 3. A Translation Example

Reasoning within Single Ontology

The *BayesOWL* framework can support common ontology reasoning tasks as probabilistic inferencesg in the translated BN, for example, given a concept description *e*, it can answer queries about concept satisfiability (whether $P(e|CT) = 0$), about concept overlapping (how close *e* is to a concept *C* as $P(e|C,CT)$), and about concept subsumption (find the concept which is most similar to *e*) by defining some similarity measures such as Jaccard Coefficient (Rijsbergen 1979).

Prototype Implementation

A prototype system named *OWL2BN* (Figure 4) is currently under active construction. It takes a valid OWL on-

tology and some consistent probabilistic constraints as input and outputs a translated BN, with reasoning services provided based on BN inference methods.

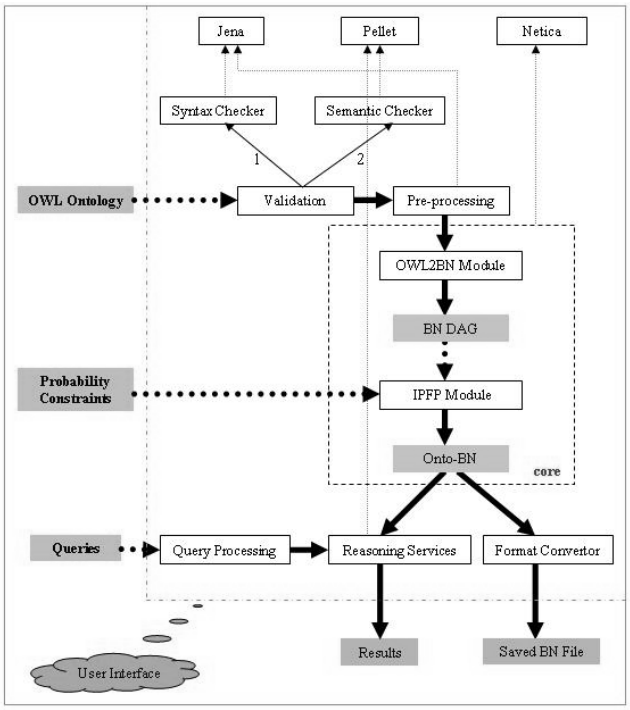


Figure 4. *OWL2BN*: Implementation of *BayesOWL*

Comparison to Related Works

Many of the suggested approaches to quantify the degree of overlap or inclusion between two concepts are based on ad hoc heuristics, others combine heuristics with different formalisms such as fuzzy logic, rough set theory, and Bayesian probability (see (Stuckenschmidt and Visser 2000) for a brief survey). Among them, works that integrate probabilities with description logic (DL) based systems are most relevant to *BayesOWL*. This includes probabilistic extensions to *ALC* based on probabilistic logics (Heinsohn 1994, Jaeger 1994); P-*SHOQ*(D) (Giugno and Lukasiewicz 2002), a probabilistic extension of *SHOQ*(D) based on the notion of probabilistic lexicographic entailment; and several works on extending DL with Bayesian networks (P-CLASSIC (Koller et al. 1997) that extends CLASSIC, PTDL (Yelland 1999) that extends TDL (Tiny Description Logic with only "Conjunction" and "Role Quantification" operators), and the work of Holi and Hyvönen (2004) which uses BN to model the degree of subsumption for ontologies encoded in RDF(S)).

The works closest to ours in this field are P-CLASSIC and PTDL. In contrast to these works, one of *BayesOWL*'s major contribution is its D-IPFP mechanism to construct CPTs from given piece-wised probability constraints. Moreover, in *BayesOWL*, by using control nodes, the "rdfs:subclassOf" relations (or the subsumption hierarchy) are separated from other logical relations, so the in-arcs to a regular concept node *C* will only come from its parent superclass nodes, which makes *C*'s CPT smaller and easier to construct than P-CLASSIC or PTDL, especially in a domain with rich logical relations.

Also, *BayesOWL* is not to extend or incorporate into OWL or any other ontology language or logics with probability theory, but to translate a given ontology to a BN in a systematic and practical way, and then treats ontological reasoning as probabilistic inferences in the translated BNs. Several benefits can be seen with this approach. It is non-intrusive in the sense that neither OWL nor ontologies defined in OWL need to be modified. Also, it is flexible, one can translate either the entire ontology or part of it into BN depending on the needs. Moreover, it does not require availability of complete conditional probability distributions, pieces of probability information can be incorporated into the translated BN in a consistent fashion. With these and other features, the cost of our approach is low and the burden to the user is minimal. We also want to emphasis that *BayesOWL* can be easily extended to handle other ontology representation formalisms (syntax is not important, semantic matters), if not using OWL.

Concept Mapping between Ontologies as an Application of BN Mapping

It is often the case when attempting to map concept *A* defined in Ontology 1 to Ontology 2 there is no concept in Ontology 2 which is semantically identical to *A*. Instead, *A* is similar to several concepts in Ontology 2 with different degree of similarity. A solution to this so-called one-to-many problem, as suggested by (Prasad, Peng, and Finin 2002) and (Doan et al. 2003), is to map *A* to the target concept *B* which is most similar to *A* by some measure. This simple approach would not work well because 1) the degree of similarity between *A* and *B* is not reflected in *B* and thus will not be considered in reasoning after the mapping; 2) it cannot handle the situation where *A* itself is uncertain; and 3) potential information loss because other similar concepts are ignored in the mapping.

With *BayesOWL*, concept mapping can be processed as some form of probabilistic evidential reasoning between the BN1 and BN2, translated from the Ontologies 1 and 2. This may allow us to address some of the aforementioned difficulties by utilizing BN techniques for integrating probabilistic knowledge and information from various sources. This section will first present a framework of variable mapping between BNs, before illustrating how ontology mapping can be conducted using this framework.

BN Mapping Framework

In applications on large, complex domains, often separate BNs describing related subdomains or different aspects of the same domain are created, but it is difficult to combine them for problem solving — even if the interdependency relations are available. This issue has been investigated in

several works, including most notably Multiply Sectioned Bayesian Network (MSBN) by Xiang (2002) and Agent Encapsulated Bayesian Network (AEBN) by Valtorta *et al.* (2002). However, their results are still restricted in scalability, consistency and expressiveness. MSBN's pair-wise variable linkages are between identical variables with the same distributions, and, to ensure consistency, only one side of the linkage has a complete CPT for that variable. AEBN also requires a connection between identical variables, but allows these variables to have different distributions. Here, identical variables are the same variables reside in different BNs.

What we need in supporting mapping concepts is a framework that allows two BNs (translated from two ontologies) to exchange beliefs via variables that are similar but not identical. We illustrate our ideas by first describing how mapping shall be done for a pair of similar concepts (*A* from ontology 1 to *B* in ontology 2), and then discussing how such pair-wise mappings can be generalized to network to network mapping. We assume the similarity information between *A* and *B* is captured by the joint distribution *P*(*A, B*).

Now we are dealing with three probability spaces: S_A and S_B for BN1 and BN2, and S_{AB} for *P*(*A, B*). The mapping from *A* to *B* amounts to determine the distribution of *B* in S_B, given the distribution *P*(*A*) in S_A under the constraint *P*(*A, B*) in S_{AB}.

To propagate probabilistic influence across these spaces, we can apply Jeffrey's rule and treat the probability from the source space as soft evidence to the target space (Pearl, 1990, Valtorta et al., 2002). The rule is given in (1), where *Q* denotes probabilities associated with soft evidence

(1) $Q(Y) = \sum_i P(Y \mid X_i) Q(X_i)$.

As depicted in Figure 5, mapping *A* to *B* is accomplished by applying Jeffrey's rule twice, first from S_A to S_{AB},, then S_{AB} to S_B. Since *A* in S_A is identical to *A* in S_{AB},, *P*(*A*) in S_A becomes soft evidence *Q*(*A*) to S_{AB} and by (1), the distribution of *B* in S_{AB} is updated to

(2) $Q(B) = \sum_i P(B \mid A_i) Q(A_i)$.

Q(*B*) is then applied as soft evidence from S_{AB} to node *B* in S_B, updating beliefs for every variable *V* in S_B by

(3) $Q(V) = \sum_j P(V \mid B_j) Q(B_j)$

$= \sum_j P(V \mid B_j) \sum_i P(B_j \mid A_i) P(A_i)$

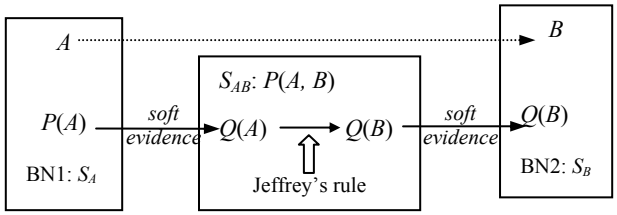


Figure 5. Mapping concept *A* to *B*

Back to the example in Figure 3, where the posterior distribution $P(Human \mid \neg Male \cap Animal)$ is (0.102, 0.898). Suppose we have another BN with a variable "Adult" with marginal distribution (0.8, 0.2). Suppose we also know that "Adult" is similar to "Human" with conditional distribution

$$P(Adult \mid Human) = \begin{pmatrix} 0.7 & 0.3 \\ 0 & 1 \end{pmatrix}.$$

Mapping "Human" to "Adult" leads to a change of latter's distribution from (0.8, 0.2) to (0.0714, 0.9286). This change can then be propagated to further update believes of all other variables in the target BN by (3).

Mapping Reduction

A pair-wise linkage as described above provides a channel to propagate belief from *A* in one BN to influence the belief of *B* in another BN. When the propagation is completed, (2) must hold between the distributions of *A* and *B*. If there are multiple such linkages, (2) must hold simultaneously for all pairs. In theory, any pair of variables between two BNs can be linked, albeit with different degree of similarities. Therefore we may potentially have $n_1 \cdot n_2$ linkages (n_1 and n_2 are the number of variables in BN1 and BN2, respectively). Although we can update the distribution of BN2 to satisfy all linkages by IPFP using (2) as constraints, it would be a computational formidable task.

Fortunately, satisfying a given probabilistic relation between *P*(*A, B*) does not require the utilization, or even the existence, of a linkage from *A* to *B*. Several probabilistic relations may be satisfied by one linkage. As shown in Figure 6, we have variables *A* and *B* in BN_1, *C* and *D* in BN_2, and probability relations between every pair as below:

$$P(C, A) = \begin{pmatrix} 0.3 & 0 \\ 0.1 & 0.6 \end{pmatrix}, P(D, A) = \begin{pmatrix} 0.33 & 0.18 \\ 0.07 & 0.42 \end{pmatrix},$$

$$P(D, B) = \begin{pmatrix} 0.348 & 0.162 \\ 0.112 & 0.378 \end{pmatrix}, P(C, B) = \begin{pmatrix} 0.3 & 0 \\ 0.16 & 0.54 \end{pmatrix}.$$

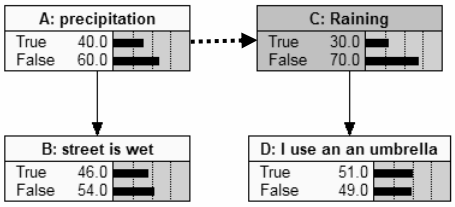


Figure 6. Mapping Reduction Example

However, we do not need to set up linkages for all these relations. As Figure 6 depicts, when we have a linkage from *A* to *C*, all these relations are satisfied (the other three linkages are thus redundant). This is because not only beliefs on *C*, but also beliefs on D are properly updated by the mapping *A* to *C*.

Several experiments with large BNs have shown that only a very small portion so fall $n_1 \cdot n_2$ linkages are needed in satisfying all probability constraints. This, we suspect, is

due to the fact that some of these constraints can be derived from others based on the probabilistic interdependencies among variables in the two BN. We are currently actively working on developing a set of rules that examine the BN structures and CPTs so that redundant linkages can be identified and removed.

Discussion and Future Work

This paper describes our ongoing research on developing a probabilistic framework for automatic ontology mapping. In this framework, ontologies (or parts of them) are first translated into Bayesian networks, then the concept mapping is realized as evidential reasoning between the two BNs by Jeffrey's rule. The probabilities needed in both translation and mapping can be obtained by using text classification programs, supported by associating to individual relevant text exemplars retrieved from the web.

We are currently actively working on each of these components. In searching for relevant exemplar, we are attempting to develop a measure of relevancy so that less relevant documents can be removed. We are expanding the ontology to BN translation from taxonomies to include properties, and develop algorithms to support common ontology-related reasoning tasks. As for a general BN mapping framework, our current focus is on linkage reduction. We are also working on the semantics of BN mapping and examine its scalability and applicability.

Future work also includes developing methods in handling inconsistent probability constraints. The study of IPFP also motivated us to develop a new algorithm named E-IPFP (Peng and Ding 2005). This algorithm is more general than the D-IPFP algorithm we used for constructing CPTs in ontology to BN translation in that it can accommodate any types of probability constraint, not only priors and pair-wise conditionals. We are working on a new algorithm that combines both E-IPFP and D-IPFP for a computationally efficient construction of CPTs for general BN.

Acknowledgement

This work was supported in part by DARPA contract F30602-97-1-0215 and NSF award IIS-0326460.

References

Bock, H. H. 1989. A Conditional Iterative Proportional Fitting (CIPF) Algorithm with Applications in the Statistical Analysis of Discrete Spatial Data. *Bull. ISI*, *Contributed Papers of 47th Session in Paris*, 1: 141-142.

Cramer, E. 2000. Probability Measures with Given Marginals and Conditionals: *I*-projections and Conditional Iterative Proportional Fitting. *Statistics and Decisions*, 18: 311-329.

Craven, M.; DiPasquo, D.; Freitag, D.; McCallum, A.; Mitchell, T.; Nigam, K.; and Slattery, S. 2000. Learning to Construct Knowledge Bases from the World Wide Web. *Artificial Intelligence*, 118(1-2): 69-114.

Csiszar, I. February 1975. *I*-divergence Geometry of Probability Distributions and Minimization Problems. *The Annuals of Probability*, 3(1): 146-158.

Deming, W. E.; and Stephan, F. F. 1940. On a Least Square Adjustment of a Sampled Frequency Table when the Expected Marginal Totals are Known. *Ann. Math. Statist.* 11: 427-444.

Ding, Z.; Peng, Y.; and Pan, R. November 2004. A Bayesian Approach to Uncertainty Modeling in OWL Ontology. In *Proceedings of 2004 International Conference on Advances in Intelligent Systems - Theory and Applications (AISTA2004)*. Luxembourg-Kirchberg, Luxembourg.

Ding, Z.; and Peng, Y. January 2004. A Probabilistic Extension to Ontology Language OWL. In *Proceedings of the 37th Hawaii International Conference on System Sciences (HICSS-37)*. Big Island, Hawaii.

Doan, A. H.; Madhavan, J.; Domingos, P.; and Halevy, A. May 2002. Learning to Map between Ontologies on the Semantic Web. In *WWW 2002*. Honolulu, Hawaii, USA.

Fukushige, Y. October 2004. Representing Probabilistic Knowledge in the Semantic Web. Position paper for *the W3C Workshop on Semantic Web for Life Sciences*. Cambridge, MA, USA.

Giugno, R.; and Lukasiewicz, T. April 2002. P-*SHOQ*(D): A Probabilistic Extension of *SHOQ*(D) for Probabilistic Ontologies in the Semantic Web. INFSYS Research Report 1843-02-06, Wien, Austria.

Heinsohn, J. 1994. Probabilistic Description Logics. In *Proceedings of UAI-94*, 311-318.

Holi, M.; and Hyvönen, E. 2004. Probabilistic Information Retrieval Based on Conceptual Overlap in Semantic Web Ontologies. In *Proceedings of the 11th Finnish AI Conference*, *Web Intelligence*, vol. 2, Finnish AI Society, Finland.

Horrocks, I.; and Sattler, U. 2001. Ontology Reasoning in the *SHOQ*(D) Description Logic. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence*.

Jaeger, M. 1994. Probabilistic Reasoning in Terminological Logics. In *Proceedings of KR-94*, 305-316.

Koller, D.; Levy, A.; and Pfeffer, A. 1997. P-CLASSIC: A Tractable Probabilistic Description Logic. In *Proceedings of AAAI-97*, 390-397.

Lacher, M.; and Groh, G. May 2001. Facilitating the Exchange of Explicit Knowledge through Ontology Mappings. In *Proceedings of the 14th International FLAIRS Conference*. Key West, FL, USA.

McCallum, A.; and Nigam, K. 1998. A Comparison of Event Models for Naive Bayes Text Classification. *AAAI-98 Workshop on "Learning for Text Categorization"*.

Mitra, P.; Noy, N. F.; and Jaiswal, A. R. 2004. OMEN: A Probabilistic Ontology Mapping Tool. In *Workshop on Meaning Coordination and Negotiation at the Third International Conference on the Semantic Web (ISWC-2004)*. Hisroshima, Japan.

Noy, N. F. 2004. Semantic Integration: A Survey Of Ontology-Based Approaches. *SIGMOD Record*, Special Issue on Semantic Integration, 33 (4).

Pearl, J. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufman, San Mateo, CA.

Pearl, J. 1990. Jeffery's Rule, Passage of Experience, and neo-Bayesianism. In H.E. et al. Kyburg, Jr., editor, *Knowledge Representation and Defeasible Reasoning*, 245-265.

Peng, Y.; and Ding, Z. July 2005. *Modifying Bayesian Networks by Probability Constraints*. Submitted to UAI 2005. Edinburgh, Scotland.

Prasad, S.; Peng, Y.; and Finin, T. July 2002. A Tool For Mapping Between Two Ontologies Using Explicit Information. In *AAMAS '02 Workshop on Ontologies and Agent Systems*. Italy.

Stuckenschmidt, H.; and Visser, U. 2000. Semantic Translation based on Approximate Re-classification. In *Proceedings of the Workshop "Semantic Approximation, Granularity and Vagueness", KR'00*.

Valtorta, M.; Kim, Y.; and Vomlel, J. 2002. Soft Evidential Update for Probabilistic Multiagent Systems. *International Journal of Approximate Reasoning*, 29(1): 71-106.

van Rijsbergen, C. J. 1979. *Information Retrieval*. London: Butterworths. Second Edition.

Vomlel, J. December 1999. Methods of Probabilistic Knowledge Integration. PhD Thesis, Department of Cybernetics, Faculty of Electrical Engineering, Czech Technical University.

Xiang, Y. 2002. *Probabilistic Reasoning in Multiagent Systems: A Graphical Models Approach*. Cambridge University Press.

Yelland, P. M. August 1999. Market Analysis Using a Combination of Bayesian Networks and Description Logics. Sun Microsystems Technical Report TR-99-78.

A Schema-Based Approach Combined with Inter-Ontology Reasoning to Construct Consensus Ontologies

Jingshan Huang, Rosa Laura Zavala Gutiérrez, Benito Mendoza García, and Michael N. Huhns

Computer Science and Engineering Department, University of South Carolina, Columbia, SC 29208

{huang27, zavalagu, mendoza2, huhns}@engr.sc.edu

Abstract

As the Semantic Web gains attention as the next generation of the Web, the issue of reconciling different views of independently developed and exposed data sources becomes increasingly important. Ontology integration serves as a basis for solving this problem. In this paper, we describe an approach to construct a consensus ontology from numerous, independently designed ontologies. Our method has the following features: i) the matching is carried out at the schema level; ii) the alignment of the ontologies is performed without previous agreement on the semantics of the terminology used by each ontology; iii) both the linguistic and the contextual features of an ontology concept are considered; iv) WordNet is incorporated into the linguistic analysis phase; v) heuristic knowledge is integrated into the contextual analysis phase; and vi) reasoning rules based on the domain-independent relationships *subclass*, *superclass*, *equivalentclass*, *sibling*, and each ontology concept's property list are used to infer new relationships among concepts. We describe a set of experiments and provide an evaluation of the results that shows the accuracy of our system.

1. Introduction

A major goal of the envisioned Semantic Web is to provide an environment where data can be shared and processed by automated tools as well as by people (Berners-Lee, Hendler, and Lassila 2001). Suppose a user wants to compare information, e.g., price, rating, and location, of nearby daycare facilities. Such information may be on the Web, but it is not in a machine-readable form. The user would need to review and process all the data exposed in each provider's website in order to get the information needed. On the Semantic Web agents can carry out this task automatically.

The idea of intelligent software agents that freely surf the Web and make sense of the information they find and the fact that such information is organized, represented, and expressed in different ways, have created the need for developing tools and techniques in order for the agents to make use of that information. Common ontologies provide the infrastructure needed to add semantics to the data on the Web so that it can be understood by agents.

It is impractical to have a unique and global ontology that includes every concept that is or might be included as part of the Web. However, it is reasonable that there might be ontologies for specific domains and sub-domains of the Web, and even for individual Web pages. It is clear, then, that the challenge is to be able to align and use different ontologies.

In this paper, we describe **PUZZLE**, a system that constructs a consensus ontology from numerous, independently designed ontologies. Our work is an extension of (Stephens, Gangam, and Huhns 2004) where the main idea is that any pair of ontologies can be related indirectly through a *semantic bridge*, consisting of many other previously unrelated ontologies, even when there is no direct relationship between the pair.

In (Stephens, Gangam, and Huhns 2004) the main technique for semantic mapping between two ontology concepts relies on simple string and substring matching. We extend this work to incorporate: further linguistic analysis; contextual analysis based on the properties of the concepts in the ontology; extended use of WordNet (Miller 1995) to include the search of not only synonyms but also antonyms, plurals, hypernyms, and hyponyms; use of the Java WordNet Library API (JWNL 2003) for performing run time access to the dictionary, instead of having to initialize the synsets a priori; integration of heuristic knowledge into the contextual analysis phase; and reasoning rules based on the domain-independent relationships *subclass*, *superclass*, *equivalentclass*, *sibling*, and each ontology concept's property list to infer new relationships among concepts.

Existing research efforts incorporate some of these features, but none has investigated them in combination. The combination addresses the major challenges described in (Stephens, Gangam, and Huhns 2004): different terminology for similar concepts and inconsistent relationships among concepts.

Our methodology is appropriate when there are a large number of small ontologies. Furthermore, in the case where the information is available through the Web, we assume that sites have been annotated with ontologies

(Pierre 2000), which is consistent with several visions for the Web (Berners-Lee, Hendler, and Lassila 2001).

The rest of the paper is organized as follows. Section 2 briefly discusses related work in ontology matching. Section 3 gives an overview of the **PUZZLE** system, whose details are described in Section 4. Section 5 reports the experiments we conducted and analyzes the results. Section 6 concludes.

2. Related Work

A lot of research work has been carried out in ontology matching. There are two approaches to ontology matching (Rahm and Bernstein 2001): instance-based and schema-based. All of the systems mentioned below belong to the latter, except for GLUE.

GLUE (Doan et al. 2003) introduces well-founded notions of semantic similarity, applies multiple machine learning strategies, and can find not only one-to-one mappings, but also complex mappings. However, it depends heavily on the availability of instance data. Therefore, it is not practical for cases where there is not a significant number of instances or no instance at all.

For HELIOS (Castano et al. 2004), WordNet is used as a thesaurus for synonyms, hyponyms, hypernyms, and meronyms. However the thesaurus has to be initialized for each domain for which it is used. If additional knowledge or a different domain is needed then the user has to input the respective terminology interactively.

PROMPT (Noy and Musen 2000) is a tool making use of linguistic similarity matches between concepts for initiating the merging or alignment process, and then use the underlying ontological structures of the Protege-2000 environment to inform a set of heuristics for identifying further matches between the ontologies. PROMPT has a good performance in terms of precision and recall. However, user intervention is required, which is not always available in real world application.

Cupid (Madhavan, Bernstein, and Rahm 2001) combines linguistic and structural schema matching techniques, as well as the help of a precompiled dictionary. But it can only work with a tree-structured ontology instead of a more general graph-structured one, which introduces many limitations to its application, because a tree cannot represent multiple-inheritance, an important characteristic in ontologies.

COMA (Do and Rahm 2002) provides an extensible library of matching algorithms, a framework for combining results, and an evaluation platform. According to their evaluation, COMA performs well in terms of precision, recall, and overall measures. Although it is a composite schema matching tool, COMA does not integrate reasoning and machine learning techniques.

Similarity Flooding (Melnik et al. 2002) utilizes a hybrid matching technique based on the idea that similarity spreading from similar nodes to the adjacent neighbors. Before a fix-point is reached, alignments between nodes are refined iteratively. This algorithm only considers the simple linguistic similarity between node names, leaving behind the node property and inter-node relationship.

S-Match (Giunchiglia, Shvaiko, and Yatskevich 2004) is a modular system into which individual components can be plugged and unplugged. The core of the system is the computation of relations. Five possible relations are defined between nodes: equivalence, more general, less general, mismatch, and overlapping. Giunchiglia et al. claim that S-Match outperforms Cupid, COMA, and SF in measurements of precision, recall, overall, and F-measure. However, as Cupid does, S-Match uses a tree-structured ontology.

In (Williams, Padmanabhan, and Blake 2003), a method is investigated for agents to develop local consensus ontologies to help in communications within a multiagent system of B2B agents. They show the potential brought by local consensus ontologies in improving how agents conduct B2B Web service discovery and composition. They also explore the influence of a lexical database in ontology merging.

3. Overview of Our Solution

The most important differences between **PUZZLE** and the systems mentioned in Section 2 are that **PUZZLE**:

- Requires no user intervention and is automated;
- Represents an ontology as a graph instead of a tree;
- Integrates WordNet by using the JWNL API;
- Applies heuristic knowledge during linguistic matching;
- Reasons with additional relations during context matching.

The goal of our work is to construct a consensus ontology from numerous independently designed ontologies. The main idea of our approach is that any pair of ontologies, G_1 and G_2, can be related indirectly through a semantic bridge consisting of other previously unrelated ontologies, even when there is no direct relationship between G_1 and G_2. The metaphor is that a small ontology is like a piece of jigsaw puzzle. It is difficult to relate two random pieces of a jigsaw puzzle until they are constrained by other puzzle pieces. Furthermore, for the semantic bridge between a given pair of ontologies G_1 and G_2, the more ontologies the semantic bridge comprises, the better the semantic match between G_1 and G_2.

In order to construct a consensus ontology from a number of ontologies, we take two ontologies and merge them into a new one, then we iteratively merge the resultant ontology with each additional one. We will explain next our method for merging two ontologies.

Suppose that original ontologies are built according to OWL Full specification (W3C 2004). Internally, our system represents an ontology using a directed acyclic graph G (V, E), where V is a set of ontology concepts (nodes), and E is a set of edges between two concepts, i.e., $E = \{(u, v) \mid u$ and v belong to V and u is a *superclass* of $v\}$. In addition, we assume that all ontologies share "#Thing" as a common "built-in" root. In order to merge two ontologies, G_1 and G_2, we try to relocate each concept from

```
PUZZLE Algorithm – merge(G1, G2)
Input: Ontology G1 and G2
Output: Merged ontology G2
Begin
    new location of G1 's root = G2 's root
    for each node C (except for the root) in G1
        Parent(C) = C's parent set in G1
        for each member pi in Parent(C)
            pj = new location of pi in G2
            relocate(C, pj)
        end for
    end for
end
```

Figure 1. **PUZZLE** Algorithm

one ontology into the other. We adopt a width-first order to traverse G_1 and pick up a concept C as the target to be relocated into G_2. Consequently, C's parent set *Parent(C)* in the original graph G_1 has already been put into the suitable place(s) in the destination graph G_2 before the relocation of C itself. The pseudocode in figure 1 describes the top level procedure of our algorithm.

The *relocate* function in the above algorithm is used to relocate C into a subgraph rooted by p_j. To obtain the correct relocation, we need to consider both the linguistic feature and the contextual feature of these two concepts (described in sections 4.1. and 4.2. respectively). The pseudocode for the *relocate* function is shown in figure 2. Notice that there is a recursive call to itself within *relocate*.

```
relocate(N1, N2)
Input: nodes N1 and N2
Output: the modified structure of N2 according to information from N1
begin
    if there exists any equivalentclass of N1 in the child(ren) of N2
        merge N1 with it
    else if there exists any subclass of N1 in the child(ren) of N2
        Children(N1) = set of such subclass(es)
        for each member ci in Children(N1)
            add links from N2 to N1 and from N1 to ci
            remove the link from N2 to ci
        end for
    else if there exists any superclass of N1 in the child(ren) of N2
        Parent(N1) = set of such superclass(es)
        for each member pi in Parent (N1)
            recursively call relocate(N1, pi)
        end for
    else
        add a link from N2 to N1
    end if
end
```

Figure 2. *relocate* Function

This recursive procedure is guaranteed to terminate because the number of the nodes within a graph is finite, and the worst case is to call *relocate* repetitively until we hit a node without child.

4. Details of the PUZZLE System

When trying to match concepts, we consider both the linguistic and the contextual features. The meaning of an ontology concept is determined by its name and its relationship with other concept(s). In this paper, we assume that the linguistic factors contribute 70 percent and the contextual factors contribute 30 percent in concept matching. The former is greater than the latter, because in our experiments, the input ontologies have less contextual information. Therefore, we do not want the contextual factors to dominate in the matching process. Notice that these weight values can always be customized according to different application requirements. For example, when merging diverse ontologies, i.e., ones with rich linguistic but poor contextual information versus ones with poor linguistic but rich contextual information, appropriate weight values can be applied accordingly.

4.1. Linguistic Matching

The linguistic factor reflects how the ontology designer wants to encode the meaning of the concept by choosing a preferable name for it. Our **PUZZLE** system uses both string and substring matching techniques when performing linguistic feature matching. Furthermore, we integrate WordNet by using JWNL API in our software. In this way, we are able to obtain the synonyms, antonyms, hyponyms, and hypernyms of an English word, which is shown to increase the accuracy of the linguistic matching dramatically. In addition, WordNet performs some preprocessing, e.g., the transformation of a noun from plural form to single form.

We claim that for any pair of ontology concepts C and C', their names N_C and $N_{C'}$ have the following mutually exclusive relationships, in terms of their linguistic features.

- *anti-match*: N_C is a antonym of $N_{C'}$, with the matching value $v_{name} = 0$;
- *exact-match*: either N_C and $N_{C'}$ have an exact string matching, or they are the synonyms of each other, with the matching value $v_{name} = 1$;
- *sub-match*: N_C is either a postfix or a hypernym of $N_{C'}$, with the matching value $v_{name} = 1$;
- *super-match*: $N_{C'}$ is either a postfix or a hyponym of N_C, with the matching value $v_{name} = 1$;
- *leading-match*: the leading substrings from N_C and $N_{C'}$ match with each other, with the matching value v_{name} = length of the common leading substring/length of the longer string. For example, "active" and "actor" have a common leading substring "act", resulting in a *leading-match* value of 3/6;
- *other*.

When relocating C, we perform the linguistic matching between C and all the candidate concepts. For each candidate concept C', if an *exact-match* or a *leading-match* is found, we put C' into C's candidate *equivalentclass* list; if a *sub-match* is found, we put C' into C's candidate

subclass list; and if a *super-match* is found, we put *C'* into *C*'s candidate *superclass* list. Then we continue the contextual matching between *C* and each concept in the three candidate lists to make the final decision.

Notice that using a synonym as the candidate of *equivalentclass* is an approximate approach, because each word could have multiple senses. We are making an assumption that different ontologies deal with similar domain (otherwise it is of little significance to align them). Therefore, in most cases, it is suitable to regard one concept's synonym(s) as a possible *equivalentclass* concept. Also, the approach to put a *sub-match* concept into another's candidate *subclass* list is approximate. In some cases it is not correct, e.g., "firstname" is not a *subclass* of "name". However, this approach does provide a lot of useful information and possible correct relationships in many cases. Similarly, *leading-match* sometimes does not offer accurate help as we expect. Because we are not considering linguistic matching alone, this kind of bias brought by *leading-match* is tolerable and under control.

4.2. Contextual Matching

The context of an ontology concept *C* consists of two parts, its property list and its relationship(s) with other concept(s). We discuss this next in detail.

4.2.1. Property List Matching

Considering the property lists, *P(C)* and *P(C')*, of a pair of concepts *C* and *C'* being matched, our goal is to calculate the similarity value $v_{Property}$ between them.

$$v_{Property} = w_{required} * v_{required} + w_{non\text{-}required} * v_{non\text{-}required}$$

$v_{required}$ and $v_{non\text{-}required}$ are the similarity values calculated for the *required* property list and *non-required* property list respectively. $w_{required}$ and $w_{non\text{-}required}$ are the weights assigned to each list. In this paper, we choose 0.7 and 0.3 for $w_{required}$ and $w_{non\text{-}required}$. $v_{required}$ and $v_{non\text{-}required}$ are calculated by the same procedure. We will explain next in detail how to obtain $v_{required}$, and from this point on, "property" means "required property" for concision purpose.

Suppose the number of properties in two property lists, P_1 and P_2, is n_1 and n_2 respectively. Without loss of generality, we assume that $n_1 \leq n_2$. There are three different matching models between two properties.

1. *total-match*
   - The linguistic matching of the property names results in either an *exact-match*, or a *leading-match* with $v_{name} \geq 0.9$; and
   - The data types match exactly.

   Let v_1 = number of properties with a *total-match*, and $f_1 = v_1/n_1$. Here f_1 is a *correcting* factor embodying the integration of heuristic knowledge. We claim that between two property lists, the more pairs of properties being regarded as *total-match*, the more likely that the remaining pairs of properties will also hit a match as long as the linguistic match between their names is above a certain threshold value. For example, assume that both P_1 and P_2 have ten properties. If there are already nine pairs with a *total-match*, and furthermore, if we find out that the names in the remaining pair of properties are very similar, then it is much more likely that this pair will also have a match, as opposed to the case where only one or two out of ten pairs have a *total-match*.

2. *name-match*
   - The linguistic matching of the property names results in either an *exact-match*, or a *leading-match* with $v_{name} \geq 0.9$; but
   - The data types do not match.

   Let v_2 = number of properties with a *name-match*, and $f_2 = (v_1 + v_2)/n_1$. Similarly to f_1, f_2 also serves as a *correcting* factor.

3. *datatype-match*

   Only the data types match. Let v_3 = number of properties with a *datatype-match*.

According to the above definition, first, we try to find out all pairs of *total-match* and filter them out of the original properties, then in the remaining properties find out all pairs of *name-match* and filter them too, and finally in the rest of original properties find out all pairs of *datatype-match*. Now we can calculate the similarity value $v_{required}$ between the two property lists.

$$v_{required} = (v_1 * w_1 + v_2 * (w_2 + w_2' * f_1) + v_3 * (w_3 + w_3' * f_2))/n_1$$

where:

- the value range of $v_{required}$ is from 0 to 1;
- w_i (i from 1 to 3) is the weight assigned to each matching model. We use 1.0 for *total-match*, 0.8 for *name-match*, and 0.2 for *datatype-match*;
- w_i'(i from 2 to 3) is the *correcting* weight assigned to the matching models of *name-match* and *datatype-match*. We use 0.2 and 0.1 respectively;

Notice that all the thresholds and arguments in the formulas mentioned in this section are based on trial-and-error.

4.2.2. Relationships among Concepts

Given any two ontology concepts, we can have the following five mutually exclusive relationships between them:

- *subclass*, denoted by $\subseteq$
- *superclass*, denoted by $\supseteq$
- *equivalentclass*, denoted by $\equiv$

- *sibling*, denoted by $\approx$ and
- other, denoted by $\neq$

OWL Full provides eleven axioms (W3C 2004): *subClassOf*, *equivalentClass*, *disjointWith*, *sameIndividualAs*, *differentFrom*, *subPropertyOf*, *equivalentProperty*, *inverseOf*, *transitiveProperty*, *functionalProperty*, and *inverseFunctionalProperty*. The first two axioms will be used to represent the *subclass-superclass* and *equivalentclass* relationships respectively.

4.3. Reasoning Rules

Based on the linguistic and contextual features, **PUZZLE** uses three domain-independent rules, each regarding the relationship among ontology concepts, to incorporate the reasoning into our system. These rules are applied to concepts from different ontologies. Therefore, we refer to them as *inter-ontology reasoning*.

Suppose we have three ontologies A, B, and C, each of which is designed according to the OWL Full specification. Furthermore, let *n(A)*, *n(B)*, and *n(C)* be the sets of concepts in A, B, and C respectively, with $n_i(A)$, $n_j(B)$, and $n_k(C)$ be the individual concept for each set (*i* from 1 to $|n(A)|$, *j* from 1 to $|n(B)|$, and *k* from 1 to $|n(C)|$), and $P(n_i(A))$, $P(n_j(B))$, and $P(n_k(C))$ be the property list for each individual concept.

Consider the property lists $P(n_i(A))$ and $P(n_j(B))$, let s_i and s_j be the set size of these two lists. There are four mutually exclusive possibilities for the relationship between $P(n_i(A))$ and $P(n_j(B))$:

- $P(n_i(A))$ and $P(n_j(B))$ are consistent with each other if and only if

  i. *Either* $s_i = s_j$ *or* $|s_i - s_j|/(s_i + s_j) \leq 0.1$, and
  ii. $v_{Property} \geq 0.9$

  We denote the corresponding concepts $n_i(A)$ and $n_j(B)$ by $n_i(A) \overset{p}{\longleftrightarrow} n_j(B)$;

- $P(n_i(A))$ is a subset of $P(n_j(B))$ if and only if

  i. $s_i \leq s_j$, and
  ii. $v_{Poperty} \geq 0.9$

  We denote the corresponding concepts $n_i(A)$ and $n_j(B)$ by $n_i(A) \overset{p}{\longrightarrow} n_j(B)$;

- $P(n_i(A))$ is a superset of $P(n_j(B))$ if and only if

  i. $s_i \geq s_j$, and
  ii. $v_{Property} \geq 0.9$

  We denote the corresponding concepts $n_i(A)$ and $n_j(B)$ by $n_i(A) \overset{p}{\longleftarrow} n_j(B)$;

- $P(n_i(A))$ and $P(n_j(B))$ have other relationship which will not be considered in our system.

Rule 1 and 2 consider two ontologies, A and B.

[**Rule 1**] This rule is straightforward, claiming that the *superclass/subclass* relationship of a class is transferable to its equivalent class(es).

- Preconditions:
  $n_i(A) \equiv n_k(B)$ and ($n_i(A) \subseteq n_j(A)$ or $n_i(A) \supseteq n_j(A)$)
- Conclusion:
  $n_k(B) \subseteq n_j(A)$ or $n_k(B) \supseteq n_j(A)$

[Rule 2] If two classes share the same parent(s), then their relationship is one of: *equivalentclass*, *superclass*, *subclass*, and *sibling*. For example, if we know that two classes have similar names and similar property lists, we still cannot conclude that they must be equivalent to each other, because of the possibility of badly designed ontologies. However, if we also know that these two classes have the same parent(s), then the probability of them being equivalent will increase substantially.

- Preconditions:
  $n_{i1}(A) \supseteq n_{i2}(A)$ and $n_{k1}(B) \supseteq n_{k2}(B)$ and
  $n_{i1}(A) \equiv n_{k1}(B)$ and
  1. $n_{i2}(A) \overset{p}{\longleftrightarrow} n_{k2}(B)$ and (the names of $n_{i2}(A)$ and $n_{k2}(B)$ have either an *exact-match*, or a *leading-match* with $v_{name} \geq 0.8$)
  2. $n_{i2}(A) \overset{p}{\longrightarrow} n_{k2}(B)$ and the name of $n_{k2}(B)$ is a *sub-match* of the name of $n_{i2}(A)$
  3. $n_{i2}(A) \overset{p}{\longleftarrow} n_{k2}(B)$ and the name of $n_{k2}(B)$ is a *super-match* of the name of $n_{i2}(A)$
  4. None of above three holds
- Conclusion:
  1. $n_{i2}(A) \equiv n_{k2}(B)$
  2. $n_{i2}(A) \supseteq n_{k2}(B)$
  3. $n_{i2}(A) \subseteq n_{k2}(B)$
  4. $n_{i2}(A) \approx n_{k2}(B)$

Rule 3 considers three ontologies, A, B, and C.

[Rule 3] If two classes have no direct relationship between them, we will refer to a third one, in order to find out the semantic bridge between the original two. In theory, the more ontologies the semantic bridge comprises, the more likely we can succeed in discovering the hidden relationships that are not obvious originally.

- Preconditions:
  $n_{i1}(A) \equiv n_{j1}(C)$ and $n_{j2}(C) \equiv n_{k2}(B)$ and
  $n_{k1}(B) \subseteq n_{k2}(B)$ and $n_{j1}(C) \subseteq n_{j2}(C)$ and
  1. $n_{i1}(A) \overset{p}{\longleftrightarrow} n_{k1}(B)$ and (the names of $n_{i1}(A)$ and $n_{k1}(B)$ have either an *exact-match*, or a *leading-match* with $v_{name} \geq 0.8$)
  2. $n_{i1}(A) \overset{p}{\longrightarrow} n_{k1}(B)$ and the name of $n_{k1}(B)$ is a *sub-match* of the name of $n_{i1}(A)$

3. $n_{i1}(A) \xleftarrow{p} n_{k1}(B)$ and the name of $n_{k1}(B)$ is a *super-match* of the name of $n_{i1}(A)$
4. None of above three holds

- Conclusion:
  1. $n_{i1}(A) \equiv n_{k1}(B)$
  2. $n_{i1}(A) \supseteq n_{k1}(B)$
  3. $n_{i1}(A) \subseteq n_{k1}(B)$
  4. $n_{i1}(A) \approx n_{k1}(B)$

5. Evaluation and Discussion of Our Results

Using a set of local ontologies designed by students, we evaluated our system in terms of precision, recall, and merging convergence. The purpose of the evaluation was to determine whether or not **PUZZLE** generates a consensus ontology.

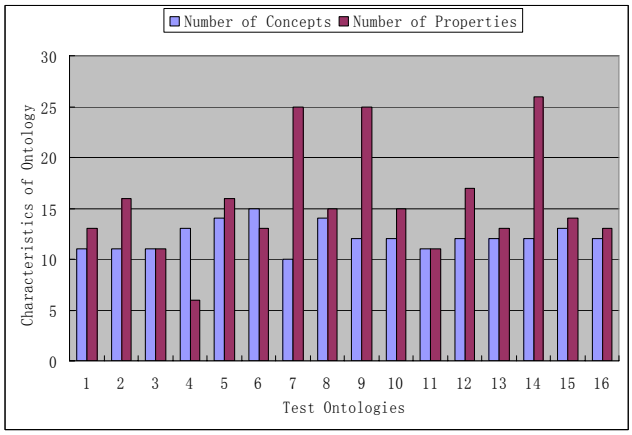


Figure 3. Characteristics of the Test Ontologies

5.1. Experimental Setup

- Configuration of the experimental platform
  Pentium 4 1.8GHz processor/512 MB RAM/40 GB hard disk/Windows XP Professional 2002 with SP2
- Programming environment
  JBuilder 9.0 with J2SE 1.5.0
- Test ontologies
  Sixteen ontologies for the domain of "Building" were constructed by graduate students in computer science and engineering at our university and used for evaluating the performance of the **PUZZLE** system. The characteristics of these ontology schemas can be found in figure 3. They had between 10 and 15 concepts with 6 to 26 properties.

5.2. Experimental Results and Analysis

Our experiments simulate having sixteen agents, each of which has a local ontology and is willing to communicate with the other agents. They try to reconcile their local ontologies to form a consensus one.

5.2.1. Evaluation of the Resultant Ontology

To decide whether a consensus ontology is obtained, we asked two ontology experts to carry out a manual mapping and we compared their results with ours. A random order was chosen during the process of merging ontologies one at a time, and both human and our system carried out the merging according to that same order, then both *precision* and *recall* measurements were applied in the evaluation. These two measurements refer to the total number of concepts with relationship of *subclass*, *superclass*, *equivalentclass*, and *sibling* up to the point at the end of each round of merging. The evaluation result is shown in figure 4. Notice that this result is not statistically valid but indicative. Both measurements reflect a promising result, except when we merged the third and the ninth ontologies. We checked the original ontologies and found out that a reason for the unsatisfactory result is due to unreasonably designed ontologies. For example, in one of the ontologies, "HumanBeing" and "InsectSpecie" are the only properties of the concept "LivingThing".

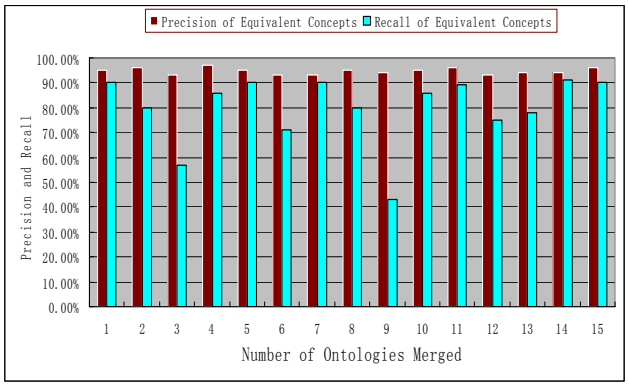


Figure 4. Precision and Recall Measurements of Resultant Ontology

5.2.2. Analysis of Merging Convergence

One hypothesis is that as each additional ontology is merged into a consensus one, there should be fewer new items (concept, relationship, or property) added to the consensus. To test this hypothesis, the following experiment has been conducted. We calculated the number of newly discovered information when the first, second, fifth, tenth, twelfth, thirteenth, and fifteenth ontologies were merged. Figure 5 shows the results of this experiment, which verifies the hypothesis.

Out of the 16 ontologies we had available for our experiments, we considered all possible combinations of the order by which they could be merged, in order to remove any bias that might be introduced by the presence of unusual ontology samples. This is a huge number; for example, there are 1680 combinations when the second ontology is to be merged, and 25000 for the fifth one. It is impossible to try all these orders. Our solution is that if the population size is less than or equal to 30 we try all

possible orders, otherwise we randomly choose a sample space of size 30.

A monotonically decreasing pattern is shown in figure 5. As the number of ontologies already merged increases, the number of concepts, relationships, and properties learned from additional ontologies decreases. We believe that the number of new items will eventually converge to zero, although the sixteen ontologies we have available for this experiment are not enough to verify this belief.

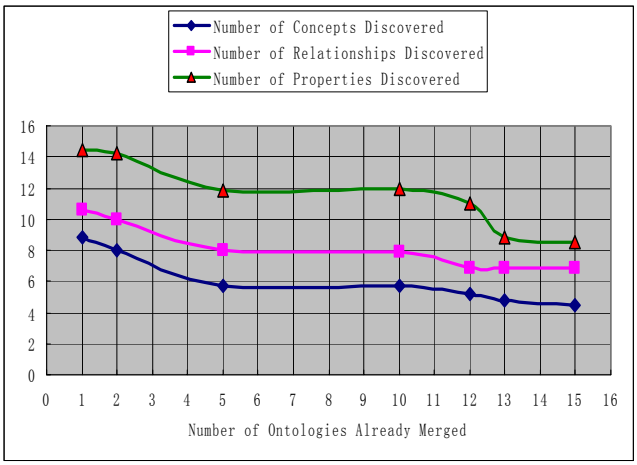


Figure 5. Merging Convergence Experiment

5.2.3. Other Features of PUZZLE

- **PUZZLE** removes redundant *is-a* links that are already specified by the transitivity of the *superclass-subclass* relationship.
- Our use of WordNet increases the accuracy. For instance, none of the original ontologies mentioned the relationship between the concepts "Monument" and "Structure". However, **PUZZLE** found out that the concept "Monument" is a *subclass* of the concept "Structure", which is quite reasonable and is an additional piece of information added to the merged ontology.

6. Conclusion and Future Work

Ontology matching is a critical operation in the Semantic Web. In this paper, we presented the **PUZZLE** system, a schema-based approach combined with inter-ontology reasoning, which reconciles ontologies for applications within a single domain. This completely automated matching is carried out at the schema level, without a previous agreement over the different terminology semantics. **PUZZLE** considers both linguistic and contextual features of an ontology concept, integrates heuristic knowledge with several matching techniques, and incorporates the reasoning among ontologies. A set of experiments showed a promising result from this system. Note that the resultant ontology represents a consensus model of a domain, but not necessarily a correct model. The possible incorrectness comes from the unreasonable (wrong) design of original ontologies.

Several remaining tasks are envisioned. We plan to adopt machine learning techniques to obtain more accurate results; take into consideration other relationships such as *partOf*, *hasPart*, *causeOf*, and *hasCause*; integrate the OWL Validator into our system; analyze the time complexity of the algorithm; and test our system against other well-known ones in ontology matching, by using more general ontology libraries.

Acknowledgements

Thanks to Dr. Larry M. Stephens for providing the source code and ontology files from a prior project; thanks to Prof. Ronald D. Bonnell and Dr. Csilla Farkas for encouraging their students to participate in the experiment. We also thank Goradia Hrishikesh and Jiangbo Dang for discussions about the system.

References

Stephens, L.; Gangam, A.; and Huhns, M. N. 2004. Constructing Consensus Ontologies for the Semantic Web: A Conceptual Approach. *World Wide Web Journal*, Vol. 7, No. 4, pages 421 - 442: Kluwer Academic Publishers.

W3C 2004. OWL Web Ontology Language Reference. *http://www.w3.org/TR/owl-ref.*

Miller, A. G. 1995. WordNet: A Lexical Database for English. In *Communications of the ACM*, Vol. 38, No. 11, pages 39 - 41: ACM Press.

Castano, S.; Ferrara, A.; Montanelli, S.; and Racca, G. 2004. Matching Techniques for Resource Discovery in Distributed Systems Using Heterogeneous Ontology Descriptions. In *Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC04)*, Vol. 1, pages 360 - 366: IEEE Computer Society Press.

Doan, A.; Madhavan, J.; Dhamankar, R.; Domingos, P.; and Halevy, A. 2003. Learning to match ontologies on the Semantic Web. *The VLDB Journal* (2003), Vol. 12, pages 303 - 319: Springer-Verlag.

Giunchiglia, F.; Shvaiko, P.; and Yatskevich, M. 2004. S-Match: an algorithm and an implementation of semantic matching. In *Proceedings of the 1st European Semantic Web Symposium*, Vol. 3053, pages 61 - 75: Springer-Verlag.

Melnik, S.; Garcia-Molina, H.; and Rahm, E. 2002. Similarity Flooding: A Versatile Graph Matching Algorithm and its Application to Schema Matching. In

Proceedings of the 18th International Conference on Data Engineering: IEEE Computer Society Press.

Noy, N. F., and Musen, M. A. 2000. PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment. In *Proceedings of the17th National Conference on Artificial Intelligence (AAAI 2000)*: AAAI Press.

Rahm, E., and Bernstein, P. A. 2001. A survey of approaches to automatic schema matching. *The VLDB Journal* (2001), Vol. 10, pages 334 - 350: Springer-Verlag.

Do, H., and Rahm, E. 2002. COMA – A system for flexible combination of schema matching approaches. In *Proceedings of the 28th VLDB Conference*: Springer-Verlag.

Madhavan, J.; Bernstein, P. A.; and Rahm, E. 2001. Generic Schema Matching with Cupid. In *Proceedings of the 27th VLDB Conference*: Springer-Verlag.

Berners-Lee, T.; Hendler, J.; and Lassila, O. 2001. The Semantic Web. *Scientific American*, Vol. 284, No. 5, pages 34 - 43: Scientific American, Inc.

JWNL 2003. Java WordNet Library – JWNL 1.3 *http://sourceforge.net/projects/jwordnet/*.

Pierre, J. M. 2000. Practical Issues for Automated Categorization of Web Sites. In *Electronic Proceedings of ECDL 2000 Workshop on the Semantic Web* (http://www.ics.forth.gr/proj/isst/SemWeb/program.html).

Williams, A.; Padmanabhan, A.; and Blake, M. B. 2003. Local Consensus Ontologies for B2B-Oriented Service Composition. In *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, Session: Ontologies, pages 647 - 654: ACM Press.

Privacy-preserving Ontology Matching

Prasenjit Mitra, Peng Liu, Chi-Chun Pan

The Pennsylvania State University,
University Park, PA 16802
{pmitra, pliu, cpan}@ist.psu.edu

Abstract

Increasingly, there is a recognized need for secure information sharing. In order to implement information sharing between diverse organizations, we need privacy-preserving interoperation systems. In this work, we describe two frameworks for privacy-preserving interoperation systems. Ontology matching is an indispensable component of interoperation systems. To implement privacy-preserving interoperation systems, we need privacy-preserving ontology matching algorithms. In this paper, we outline frameworks for privacy-preserving ontology matching and discuss the privacy implications of the frameworks.

Introduction

Though researchers have built tools that enable organizations to share information, largely, most of these tools have not taken into the account the necessity of maintaining the privacy and confidentiality of the data and the metadata of the organizations that want to share information.

Consider the (hypothetical, but seemingly probable) scenario where the U.S. and U.K. military want to share information. They want to share data only about the mission at hand while preserving the privacy of their systems. That is, they want to share information without exposing to each other any significant details about the schema and other metadata about their systems. To the best of our knowledge, the current state-of-the-art systems do not allow privacy-preserving information sharing without sharing that is required in such a scenario.

Not only does the need for secure information sharing arise among organizations that want to share information among each other, but the need also arises for intra-organization information sharing. Large organizations, like large corporations or even the U.S. Department of Homeland Security, have a number of departments with varying levels of autonomy. That is, even within the same organization, different departments use information systems that were autonomously constructed. For example, in a large software development firm, the data center may be located at a different geographical location than the software development department, and due to their different needs, the two departments maintain different systems. The challenge of secure information sharing is prevalent even in these scenarios.

Not only must an organization preserve the privacy of its data, but it must also preserve the privacy of sensitive metadata (or meta-information). Metadata describes how data are organized in the organization (e.g., data schema), how accesses are controlled in the organization (e.g., the internal access control policy and role hierarchies), and the semantics of the data used in the organization (e.g., ontology).

Organizations seeking to interoperate are increasingly using metadata like ontologies to capture the semantics of terms used in the information sources maintained by the organizations. Traditionally, it has been assumed that these ontologies will be published by the organization. Published ontologies from different organizations are matched and matching rules generated. Queries to information sources are rewritten using these matching rules so that the vocabulary used in the query is the same as that used by the information source.

Unlike in the traditional scenario, some organizations do not want to publish their metadata or even share the metadata with external users. Yet, they want to enable interoperation. In this scenario, the privacy of the metadata, e.g., the ontologies of information sources or the schema of databases, must be preserved. That is, any user outside the host organization should not have access to the ontologies in cleartext. This is because in a mediated architecture, if the mediator is malicious or if an intruder breaks in to the mediator, substantial loss of information and privacy occurs.

In this paper, we present two frameworks for privacy-preserving interoperation and especially highlight their privacy-preserving ontology-matching components. These frameworks achieve ontology matching with minimal "privacy leak" of the ontologies being matched. The interoperation system does not assume a trusted mediator. Ideally, the organizations want the mediator to gain minimal information about the data and the metadata stored in its information sources. In our system, the mediator operates over encrypted queries, encrypted ontologies and encrypted data.

In the first ontology-matching framework, we show how totally automated ontology mapping can be achieved. In this framework, the queries and ontologies are encrypted using a symmetric private key shared between the organizations interoperating. In the second framework, we show how semi-automatic ontology mapping can be achieved. In this framework, the ontologies of each organization are encrypted using their own private keys and thus even interoperating organizations do not share their ontologies. To the best of our knowledge, there exists no existing work on privacy-preserving ontology matching.

The difficulty in preserving the privacy of the ontologies is that totally automated ontology matching does not work very well in practice (despite individual claims in research settings). Even if automated methods achieve about 70-80% accuracy, the matching rules missed by the automated matchers must be generated manually. Now, in order for a human expert to match the ontologies to generate the missing matches, the ontologies need to be exposed to the expert in cleartext. Therefore, in our second framework, we try to limit the exposure of the ontologies only to the ontology-matching expert.

Preliminaries

Ontology mapping techniques can be classified into the following categories:

1. Word Similarity Based: In this case the concepts across ontologies are matched using the similarity of the words that appear in the ontologies (Mitra, Wiederhold, Decker).
2. Structural Similarity Based: This set of algorithms use the structure of the ontologies to match the concepts in the ontologies. (Melnik, Garcia-Molina, Rahm), (Noy and Musen).
3. Instance Based: Concepts in ontologies are matched using the similarity of their instances. Among the instance-based algorithms, we can further sub-classify them into two types:
    a. *Opaque Matching*: In this case, the matching does not depend upon the values of the instances but on the statistical properties, like distribution, entropy, mutual information etc. of the instances of a concept (Kang and Naughton).
    b. *Pattern-based Matching*: In this case, the algorithm identifies patterns in the values of the instances and uses similar patterns in their values to indicate that two concepts are similar.
4. Inference Based*:* The semantics of concepts in ontologies are expressed as rules using a logical language (say, the Web Ontology Language, OWL). Using an inference engine and these ontology rules, concepts across ontologies can be matched.

There are also algorithms that use hybrid or multiple strategies (Doan et al.).

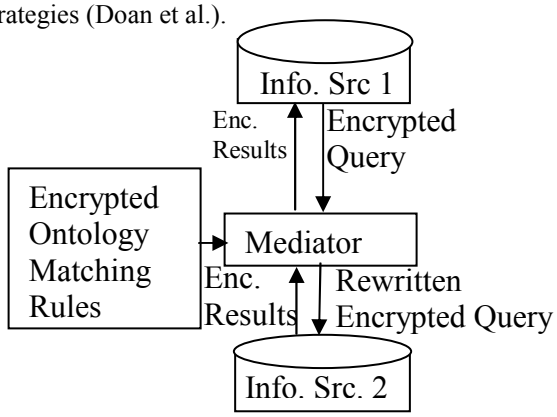


Figure 1. Privacy –preserving Interoperation System Architecture Using Private Key

Privacy-preserving Automated Ontology Matching

In this scenario, we assume that the organizations, say A and B, seeking to enable interoperation have a symmetric private key, $K_{A\text{-}B}$. The queries originating from both A and B, posed to the mediated system (as shown in Figure 1) are encrypted using the private key $K_{A\text{-}B}$. The ontology matching rules used by the mediator are also encrypted using the same key.

We look at the ontology matching algorithms used to generate the encrypted ontology matching rules. As shown in Figure 2, the input to the ontology matcher, the source ontologies corresponding to the information sources for both A and B, are encrypted using $K_{A\text{-}B}$. The automated ontology matcher operates on the encrypted ontologies to match concepts across the ontologies.

Several ontology matching algorithms use dictionaries, thesauri or corpuses of documents to identify matching concepts. In order for these algorithms to work, the dictionaries, thesauri, or corpuses should also be encrypted using the same key, $K_{A\text{-}B}$.

Structure-based ontology matching techniques work fine even if the terms and relationships of the ontologies are encrypted because either they do not depend on the terms or relationships or even if they do, as long as the same labels are similarly encrypted, these algorithms are unaffected.

Instance-based matching algorithms that are opaque work fine without any modification because the statistical properties, like distribution, frequency, entropy, mutual information, of the instance values are not changed by encrypting it, however, pattern-based matching algorithms do not work because in most encryption systems destroy the patterns in the instance values.

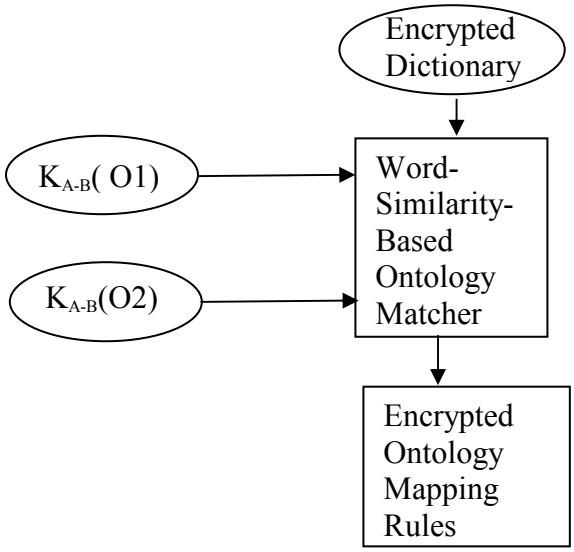


Figure2. Privacy-preserving Automated Ontology Mapping

Semi-automated Ontology Matching Framework

The framework shown above has two important drawbacks:

1. It assumes that the process of ontology matching can be totally automated. Note that in the process outlined above, the ontology matcher has access to only encrypted ontologies and cannot decrypt the ontologies. Typically, human experts cannot match encrypted ontologies. Even if they do, matching ontologies without knowing the semantics of the concepts (because they are encrypted and thus their semantics is undecipherable) will not result in very accurate match generation.
2. Because the two organizations use a shared symmetric private key, each organization can observe the communication of the other with the mediator and obtain the other organization's ontology. In cases, where the organizations do not want to share their ontologies even with the organization they are sharing information with, such an arrangement is not acceptable.

In order to remedy the above-mentioned drawbacks, we offer the following interoperation and ontology-matching framework.

Interoperation Framework

Because the mediator cannot be trusted, the queries are encrypted and sent to the mediator. In this framework, each organization has a *unique* secret private key that it uses to encrypt the queries and ontologies. The mediator uses encrypted ontology-matching rules. For example, an encrypted ontology matching rule may be

((Y InstanceOf K2(O2.Car)) &
(Z Equals Y.K2(O2.Price)) &
(Z > 40,000)
=> (Y InstanceOf K1.(O1.LuxuryCar)) (R1)

The rule above indicates that if Y is an instance of car(O2.Car), Z is the the price(O2.Price) of Y, and Z is greater than 40,000, then Y is also an instance of luxury car. The keys K2 and K1 encrypt terms in O2, O1 respectively.

Using such an encrypted rule, the mediator can rewrite a given query, e.g., (?X InstanceOf K1(O1.LuxuryCar)) that asks for all instances of luxury-car(O1.LuxuryCar), by substituting the left-handside of (R1) for the query. Note that all the ontology terms in the query and the ontology-matching rule are encrypted and thus the mediator does not have access to those terms.

Privacy-preserving Semi-automated Ontology Matching

If we intend to use a human expert in the process of ontology matching, the human expert must have access to the ontologies in cleartext because encrypted labels will make no sense to the expert. In this scenario, as shown in Figure 3, each organization encrypts the ontologies using a session key that it shares with the expert (ontology matcher). Upon receiving the ontologies to be matched, the expert decrypts the encrypted ontologies using the session key. Each organization also has a public key that it has publicized via a certifying authority. The certifying authority serves as the trusted intermediary between the expert and the organizations. The expert (using a semi-automatic ontology matcher) matches the two ontologies and then creates a set of ontology matching rules similar to the rule shown in the example above. Let us say the source ontologies being matched are O1 and O2. Terms appearing in a ontology matching rule and O1 are encrypted using the public key (K1) of the first organization and terms appearing in the ontology matching rule and O2 are encrypted using the public key (K2) of the second organization (Figure 4). The mediator rewrites a query obtained from one organization, say Org1, encrypted using its key K1, to a query where all terms are from another organization's, say Org2, ontology, encrypted using its key K2 using the ontology mapping rules.

Related Work

Clifton et al., have argued about the need for and highlighted issues in privacy-preserving data integration and sharing. Agarwal and Srikant have shown how to mine data while preserving privacy. However, to the best of our knowledge, there exists no prior research that shows how privacy-preserving ontology matching can be enabled. Our interoperation architectures have been influenced by existing works on access-control in information interoperation systems (Damiani et al., Dawson, Qian and Samarati, de Capitani di Vimercati and Samarati, Gong and Qian).

Though there does not exist work on privacy-preserving ontology matching, as discussed above, several existing works have provided algorithms for ontology matching (Melnik, Garcia-Molina, and Rahm, Noy and Musen, 2001, Doan, et al., Hovy, Euzenat and Volchev, Shvaiko,

Giunchiglia, and Yatskevich, Mitra, Wiederhold, and Decker, and Noy and Musen, Prasad, et al).

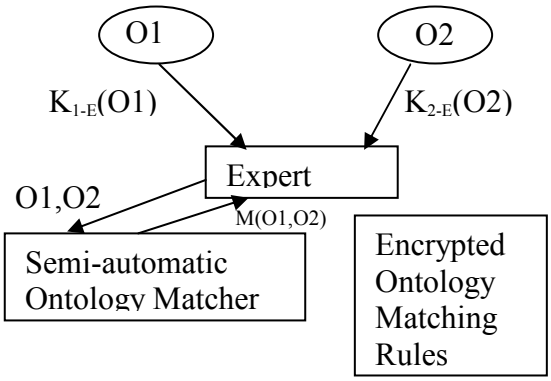


Figure 3. Privacy-preserving Semi-automatic Ontology Matching

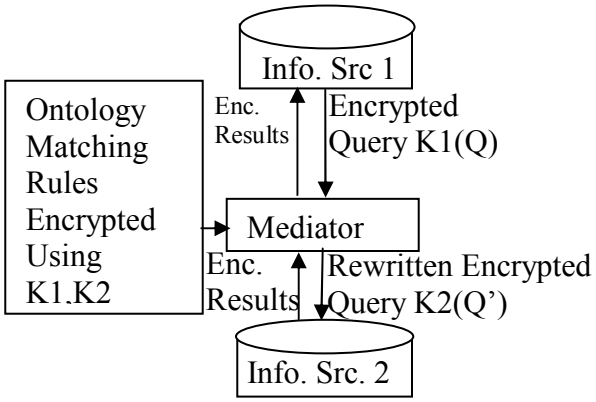


Figure 4: Privacy-preserving Interoperation Using Public Keys

Conclusion

Maintaining privacy in interoperation systems is becoming increasingly important. Ontology matching is the primary means of resolving semantic heterogeneity. Ontology matching helps establish semantic correspondence rules that are used for query rewriting and translation in interoperation systems. For information systems that want maximum privacy, the privacy of their ontologies must be maintained. In this paper, we describe two frameworks for privacy-preserving interoperation and show how we can implement privacy-preserving ontology matching.

References

Agrawal, R. and Srikant, R. Privacy-Preserving Data Mining. *In Proc. of the ACM SIGMOD, 2000.*

Clifton, C., Doan, A., Elmagarmid, A., Kantarcioglu, M., Schadow, G., Suciu, D., and Vaidya, J. Privacy Preserving Data Integration and Sharing, *Proc. of the 9th Int. Workshop on Data Mining and Knowledge Discovery, 2004 (DMKD-04)*

Damiani, E., De Capitani di Vimercati, S., Fugazza, C., and Samarati, P. Extending Policy Languages to the Semantic Web. *ICWE* 2004 330-343

Dawson, S., Qian, S., Samarati, P. Providing Security and Interoperation of Heterogeneous Systems. *Distributed and Parallel Databases*, vol. 8, no. 1, Jan. 2000, 119-145.

De Capitani di Vimercati, S., and Samarati, P. Authorization Specification and Enforcement in Federated Database Systems. *Journal of Computer Security*, vol. 5, no. 2, 1997, 155-188.

Doan, A., Madhavan, J., Domingos, P., and Halevy, A. Learning to map between ontologies on the semantic web. *In The Eleventh International WWW Conference*, Hawaii, US, 2002.

Euzenat, J., and Valtchev, P. Similarity-based ontology alignment in OWL-Lite. *In The 16th European Conference on Artificial Intelligence (ECAI-04),* Valencia, Spain, 2004.

Gong, L. and Qian, X. The Complexity and Composability of Secure Interoperation. *IEEE Symp. Security and Privacy*, (Oakland, CA, USA. 1994).

Hovy, E. Combining and standardizing largescale, practical ontologies for machine translation and other uses. *In The First International Conference on Language Resources and Evaluation (LREC)*, pages 535–542, Granada, Spain, 1998.

Kang, J., Naughton, J. F.: On Schema Matching with Opaque Column Names and Data Values. Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD), San Diego, California, June 2003.

Melnik, S., Garcia-Molina, H. and Rahm, E. Similarityflooding: A versatile graph matching algorithm and its application to schema matching. *In 18th International Conference on Data Engineering (ICDE-2002)*, San Jose, California, 2002. IEEE Computing Society.

Mitra, P. and Wiederhold, G. Resolving terminological heterogeneity in ontologies. *In Workshop on Ontologies and Semantic Interoperability at the 15th European Conference on Artificial Intelligence (ECAI)*, July 2002.

Mitra, P., Wiederhold, W., and Decker, S. A scalable framework for interoperation of information sources. *In The 1st International Semantic Web Working Symposium (SWWS'01),* Stanford University, Stanford, CA, 2001.

Noy, N.F., and Musen, M.A. Anchor-PROMPT: Using non-local context for semantic matching. *In Workshop on Ontologies and Information Sharing at the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI-2001)*, Seattle, WA, 2001.

Noy, N. F., and Musen, M. A.. The PROMPT suite: Interactive tools for ontology merging and mapping. *International Journal of Human-Computer Studies*, 59(6):983–1024, 2003.

Prasad, S., Peng, Y., and Finin, T. A tool for mapping between two ontologies using explicit information. *In AAMAS 2002 Workshop on Ontologies and Agent Systems,* Bologna, Italy, 2002.

Shvaiko, P., Giunchiglia, F. and Yatskevich, M. S-Match: an Algorithm and an Implementation of Semantic Matching, *Proc. of the 1st European Semantic Web Symposium, 2004 (ESWS-04).*

Contexts in Dynamic Ontology Mapping

Paolo Besana and **Dave Robertson**
Centre for Intelligent System and their Applications
School of Informatics
University of Edinburgh

Abstract

Agents in open systems interact continuously, each possibly having a different ontology. Mapping in advance all the ontologies that an agent can encounter is not feasible, as all the possible combinations cannot be foreseen. Mapping complete ontologies at run time is a computationally expensive task. This paper proposes a framework in which mappings between terms may be hypothesised dynamically as the terms are encountered during interaction. In this way, the interaction itself defines the context in which small, relevant portions of ontologies are mapped. We use this way of scoping the ontology mapping problem in order to apply mapping heuristics in a more focused way.

Problem description

In order to act properly after receiving a message from an external entity, an agent must understand the content of the message.

A message is created by mapping concepts in the sender's representation of the domain into the terms that compose the message, conforming to the syntax of the language it uses. The receiver maps the terms in the message to the concepts in his own representation, helped by the syntax rules that structure the message. If a term is mapped to a different concept by the receiver agents, or cannot be mapped, then a misunderstanding arises.

The problem would not exist if both agents shared the same representation of the domain, but this is not the most common case.

For example, in an open B2B market, agents gather to offer and to request services or products, working as proxy for their companies. Agents converge to the market from different backgrounds, and are likely to have different ontologies. The brokers receive advertisements of offers, and must classify them correctly to match them with the requests. As agents continuously arrive to the market and leave, the number of possible combinations of agents is high, .

Common approach

Early attempts to overcome the heterogeneity in the representations were to develop general ontologies that could cover a majority of domains and could be shared by the agents. This approach has been unsuccessful on a large scale. First, it proved difficult to find an agreement on what ontology to use. Second, it is difficult to manage the evolution of the ontology: an old version can be inconsistent with a newer one (Hameed, Preece, & Sleeman 2003).

More recent attempts are instead focused on reconciling different ontologies, allowing their coexistence.

Problems with Ontology Mapping in MAS

Most mapping processes are aimed at statically aligning complete ontologies (Kalfoglou & Schorlemmer 2003; Giunchiglia, Shvaiko, & Yatskevich 2004; Nuno & Rocha 1999): two or more ontologies are reconciled and the result is stored for future use.

Preparing in advance all the possible mappings between the ontologies is not feasible in open multi-agent systems, as it is impossible to foresee all the combinations of agents involved in the interactions. Mapping whole ontologies, often a lengthy process, may not be feasible at real time. Interactions should be quick and many can occur at the same time. Moreover, only portions of ontologies may match, as agents can have ontologies about different domains.

Proposed approach

A complete agreement over the semantics is not required in MAS: agents interact only when they must, and they need to understand each other just enough to perform their task. Once the task is performed, mutual understanding is no longer important: agents in an open system interact continuously with different agents, and the mapping found once might be useless any other time.

To perform a coordinated task, the involved agents need to share only the parts of their knowledge contextual to the interaction. It is possible to exploit this idea and map dynamically and only when needed the portions of the ontologies required for the context of the interaction.

Definitions and assumptions

Agent model

Each agent a_i has its own communication environment e_i, consisting of the ontology O_i that defines the terms used by the agent and of the axioms it can use to reason.

An environment can be seen as the context of an agent, as described in (Giunchiglia 1992), but renamed to avoid name conflicts with the concept of context used in this paper.

Any definition is valid only within an environment. An agent can reason over concepts defined in other environments only if mapped to its own concepts.

An agent, for the sake of this paper, can be model as being composed by two layers: a *communication layer*, and a *reasoning layer*. The communication layer is the interface between an agent and the other agents in the system. In the basic case, it handles the transmission and reception of messages. The reasoning layer contains all the agent's skills and knowledge, and it is accessible from the communication layer through access points.

Communication model

During an interaction k an agent a_i sends to an agent a_j a message m composed of terms. For brevity, terms defined in the agent's environment will be called *internal terms*, and will be referenced as w_i(see figure 1), while terms defined in other environments will be called *external terms*, and will be referenced as t_i.

For an interaction k, every involved agent a_i publishes an ontology subset O_{ki}, valid in the context of the interaction, to explain the terms it has used in the messages.

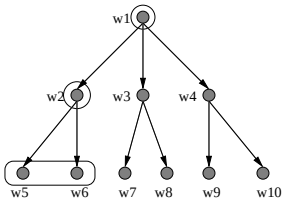


Figure 1: O_i ontology

Semantic bridges

The semantic relations between terms defined in different environments is defined in semantic bridges (Nuno & Rocha 1999). A bridge b is the tuple:

$$b = \langle relation, t, w, c(true), c(false) \rangle$$

where $relation$ can be equivalence or subsumption, while $c(true)$ is the confidence level that the bridge is correct and $c(false)$ that the bridge is wrong.

A bridge b_h is more generic ($\succeq$) than another bridge b_g, if the external term t is the same in both and the internal term w_h of b_h subsumes the internal term w_g in b_g:

$$b_h \succeq b_g \leftrightarrow (w_h \sqsupseteq w_g) \wedge (t_h = t_g)$$

Conversely b_h is more specific than b_g if w_g subsumes w_h:

$$b_h \preceq b_g \leftrightarrow (w_h \sqsubseteq w_g) \wedge (t_h = t_g)$$

During the interaction k the bridges are stored in the set B_k and are used to translate the calls to the reasoning layer.

Framework

An external term t is mapped only when encountered during an interaction. Initially it can only guess at the proper bridges between t and the terms in its own ontology. These hypotheses must be verified, and the most likely is kept.

Without any *a priori* knowledge any possible mapping could be the correct one, and the hypotheses cover all the possible bridges between the external term t and the terms w_i. To make on-the-fly mapping feasible, the number of hypotheses should be drastically reduced.

In the framework, unlikely hypotheses are pruned by the *filter* elements using heuristics based on the experience of past interactions and on the context of the current interaction. The filters aim to minimise, on average, the number of wrong hypotheses to check.

Once the hypotheses are filtered, the *rule* elements generate an arguments in favour or against a hypothesis. The arguments are then combined by the framework to give an overall confidence level for the hypothesis.

Rules can exploit algorithms developed for static mapping, such as S-Match (Giunchiglia, Shvaiko, & Yatskevich 2004), to compute the matching, as the number of useless mappings to verify is reduced by filters.

A generated *argument* is a proposition coupled with two degrees of confidence, one that the proposition is true, and one that the proposition is false. Arguments are organised in a tree: the root is the hypothesis to verify, supported or attacked by the arguments in the nodes. An argument may recursively need other arguments to support it.

Framework explained

The mapping process is iterative. At every iteration i a semantic bridge b_{tki}, more specific than the bridge $b_{tk(i-1)}$ from the previous iteration, is created for the term t:

$$b_n \preceq b_{n-1} \preceq \ldots \preceq b_1$$

At every iteration i the function executes three steps:

- generates hypotheses,
- filters hypotheses and keeps the most probable ones,
- collects evidence for the remaining hypotheses, selects the most reliable hypothesis.

The loop ends when it becomes impossible to generate hypotheses that imply those proved in the previous step, or none of the hypotheses generated can be proved. The bridge created in the last iteration, and therefore the most specific, is returned and added to the set B_k.

Generate the hypotheses

At this step of each iteration i, the system receives the external term t and the mapping $b_{tk(i-1)}$ proved in the previous iteration, and returns a set of hypotheses Ω about the most generic mappings that imply $b_{tk(i-1)}$.

For example, if $b_{tk1} = \langle t \sqsubseteq w_1 \rangle$, $b_{tk2} = \langle t \sqsubseteq w_2 \rangle$, given the ontology in figure 1, then for the iteration #3:

$$\Omega_3 = \{\langle \{\sqsubseteq, \sqsupseteq, \equiv\}, t, w_5 \rangle, \langle \{\sqsubseteq, \sqsupseteq, \equiv\}, t, w_6 \rangle\}$$

Filter the hypotheses

In the this step, the system combines different filters and produces an argumentation tree for each of the hypotheses selected from the set Ω generated in the previous step:

A filter f_i is characterised by its *breadth* and its *confidence*. The first is the "band-pass" of the filter: the narrower the filter, the fewer hypotheses are left to verify. If none of the filtered hypotheses could be proved, this step is repeated and the narrowest filter used previously is removed. The second indicates how likely is it that the correct hypothesis is in the selected subset. It is used as the first argument added to the argument tree of each filtered hypothesis.

After a term is successfully mapped, the filter receives the new bridge as feedback, and uses it to improve its predictive capability.

Select the best hypothesis

In this step, the system processes the set of hypotheses trees generated by the previous step, and tries to extract the most likely one. If the system fails to select any hypothesis, it goes back to the previous step, relaxes the filter if possible, and tries to obtain a wider set of hypotheses.

This step is composed of three actions.

Collect evidence For each hypothesis the system generates arguments using rules. A rule r_i is characterised by two confidence levels, that measure how strong or weak is the support or the attack of the generated argument: $c(hp|arg)$ is the confidence that the hypothesis is true, given that the argument is true, while $c(\neg hp|\neg arg)$ is confidence that the hypothesis is false, given that the argument is false. The argument is produced by an external function, that receives as input the hypothesis and a set of information specified in the rule. The information is collected by the system, and it is relative to the terms in the hypothesis: it can be the superclass, or the subclasses or the instances of one of the term. Information about internal terms is easily accessible, while the agent may ask the information about external terms to the other agent if it is not contained in the published ontology. External information may trigger further mapping to allow reasoning over the imported terms.

Combine evidence The arguments in the tree are combined to obtain two confidence levels for the hypothesis: one that the hypothesis is true, and one that it is false.

If a hypothesis is supported by one argument, the confidence that the hypothesis is true is computed as:

$$c(hp) = c(hp|arg)c(arg)$$

where $c(hp|arg)$ is given by the rule and $c(arg)$ is computed for the argument. Similar considerations apply for the confidence of the hypothesis being false. When there is more than one argument, the confidences must be combined.

It cannot be assumed that the confidences sum to one: if a rule establishes that a hypothesis is true with 0.4 of confidence, it does not imply that the hypothesis is false with 0.6 of confidence. Therefore, the theory used to combine confidences should be able to express ignorance about the truth value of the hypothesis.

One possible approach is Dempster-Shafer theory (Yager 1994), which computes the probability of a proposition supported by evidences. Following Dempster-Shafer theory, $c(hp)$ is interpreted as the *belief* that the hypothesis is true. While $1 - c(\neg hp)$ is the *plausibility* the hypothesis is true which is the extent to which the available evidence fails to refute the hypothesis. The interval between the two values is the ignorance interval.

The theory provides a formula, called Dempster's rule of combination, to combine evidences for a proposition.

Harvest Hypothesis At the end of an iteration *i* , the hypothesis with the highest confidence is selected. In some cases there might be more than one hypothesis within a narrow band of confidence: in this case the system first tries to apply more rules - if available - to gather more evidence for the conflicting hypotheses. If no rules can be applied, the strongest hypothesis is selected. Then the procedure restarts, until no more hypotheses can be generated.

Possible models of filters

Filters

Filters must operate rapidly, making it difficult to apply complex, symbolic or inductive inference methods. Nevertheless, filters can exploit the large volume of event-based data from the interactions and determine statistical patterns threaded in the dialogues.

Statistical contexts

A possible pattern to recognise is that some terms tend to appear together in interactions: some of these terms are contextual to the topic of the conversation (*buy*, *computer*,...), while other terms are auxiliary to any kind of conversation (*ask*, *inform*,...).

Following this intuition, the terms can be clustered together, and each cluster is a possible context for an interaction. The contexts are created and updated using the feedback from the framework. The contexts are used to classify dialogues as they unfold, and to predict which are the most likely terms that can occour during a conversation. This excludes hypotheses relative to terms that have never appeared in the context.

Formal description More formally, a *context* is a triple:

$$C = \langle id, N, S \rangle$$

where N is the number of dialogues classified by the context and S is the set of internal term elements η that distinguish the context.

Each term element η_i in S is a pair:

$$\eta_i = \langle w, \mu_C \rangle$$

where w is the term in the agent's ontology and μ_C is the grade of membership of the term in the context: terms may appear in different contexts with different frequencies.

Related to the grade of membership of a term there is the function $\mu_C(K)$ that returns the grade of membership to a context C of a set K of terms:

$$\mu_C(K) = \frac{1}{|K|} \sum_{w \in K} \mu_C(w)$$

How they are used Contexts are used to classify dialogues as they are performed. Every time a new term is mapped, the system tries to classify the dialogue finding the contexts that maximise the function $\mu_C(W)$, where W is the set of the internal terms in the bridges contained in B_k.

At the beginning of the mapping process, few terms are mapped and it is difficult to classify the dialogue properly, because more than a context can do it. As the dialogue unfolds, the number of terms in W increases and the number of contexts that can classify them is reduced.

The contexts that classify the dialogue are used to filter the generated hypotheses set: if some terms in the set never appear in the contexts, then it is possible to exclude these hypotheses, adding evidence for the remaining hypotheses.

How they are created When a dialogue is finished, the internal terms W are added to the context that better classifies them. If no context classify them well enough, then a new context is created.

Past mapping experience

Another possible pattern to identify is that some external terms tend to have always the same semantic relations with the same terms in the agent's ontology.

Formal description The set of previous mappings Λ contains a tuple λ_i for each mapping proved in the past, composed of three elements:

$$\lambda_i = \langle b, s_m, n_a \rangle$$

b is the hypothesis proved in the past, s_m is the cumulative confidence of the hypothesis, and n_a is the number of time the term mapped in the hypothesis has appeared in dialogues.

How they are used When the system must select hypotheses for an external term, it can look in past mappings for the term. It then keeps the hypotheses implied by the past mappings, and discards the others.

For the ontology in figure 1, given the generated hypotheses $\Omega = \{\mathtt{t} \sqsubseteq \mathtt{w}_2, \mathtt{t} \sqsubseteq \mathtt{w}_3, \mathtt{t} \sqsubseteq \mathtt{w}_4\}$ and the set of past mappings $\Lambda = \{\langle \mathtt{t} \sqsubseteq \mathtt{w}_5, 4, 5 \rangle\}$, the filter should keep only $\mathtt{t} \sqsubseteq \mathtt{w}_2$ as the past mapping $\mathtt{t} \sqsubseteq \mathtt{w}_5$ implies it.

How they are created Mappings established for a particular external ontology and received as feedback from the framework are stored for future use. When mappings are encountered repeatedly, the confidence s_m in the past mapping λ_i is increased by the confidence in the bridge.

There is no issue about inconsistency, as conflicting past mappings are used only as suggestions about the order in which the hypotheses should be checked: conflicting hypotheses are tolerated by collecting evidence in favour or against them.

Related work

The COMA project (Do & Rahm 2002) is focused on combining different matchers to obtain a plausibility level for the computed correspondences. It introduces the reuse of past mappings, although for a different purpose. The abstraction of the argumentation tree in this paper subsumes the distinction between simple matchers and hybrid matchers. The QOM project (Ehrig & Staab 2004) addresses the trade off between efficiency and quality, introducing the concept of filtering the mapping candidates before verifying them with similarity comparators. However, the filtering is based only on properties of the ontologies (labels of nodes or hierarchy): it is does not exploit gained experience and it is not concerned about the purpose of the mapping as a mean to prune useless candidates.

In fact, both projects are oriented toward mapping whole ontologies, without any reference to interactions and their contexts.

Conclusion

In this paper we proposed a framework that allows agents with different ontologies to interact in order to perform a task. The mutual understanding during the interaction is reached dynamically mapping the ontologies. The framework exploits both the structure of the ontologies and statistical patterns threaded in the dialogues to produce heuristics used to improve the work of standard mapping algorithms.

The first advantage is that there is no need to foresee and map in advance all possible combinations of ontologies, because mappings take place only when needed. The second is that part of the algorithms developed and tested for this task can still be used.

This research is at a very early stage, and many details still need to be studied in depth.

References

Do, H. H., and Rahm, E. 2002. Coma - a system for flexible combination of schema matching approaches. In *VLDB*, 610–621.

Ehrig, M., and Staab, S. 2004. Qom - quick ontology mapping. In *International Semantic Web Conference*, 683–697.

Giunchiglia, F.; Shvaiko, P.; and Yatskevich, M. 2004. S-match: an algorithm and an implementation of semantic match. In *In Proceeding of the European Semantic Web Symposium*, 61–75.

Giunchiglia, F. 1992. Contextual reasoning. Technical report, IRST, Istituto per la Ricerca Scientifica e Tecnologica.

Hameed, A.; Preece, A.; and Sleeman, D. 2003. *Ontology Reconciliation*. Germany: Springer Verlag. 231– 250.

Kalfoglou, Y., and Schorlemmer, M. 2003. Ontology mapping: the state of the art. *Knowledge Engineering Review*.

Nuno, S., and Rocha, J. 1999. Mafra - an ontology mapping framework for the semantic web. In *Proc. of the 13th European Conf. on Knowledge*.

Yager. 1994. *Advances in the Dempster-Shafer Theory of Evidence*. John Wiley, New York.

Measuring Similarity of Elements in OWL DL Ontologies

Thanh-Le Bach, Rose Dieng-Kuntz

ACACIA Project, INRIA Sophia Antipolis
2004 route des Lucioles, BP 93, 06902 Sophia Antipolis, France
{Thanh-Le.Bach, Rose.Dieng}@sophia.inria.fr
http://www-sop.inria.fr/acacia/acacia.html

Abstract

OWL becomes nowadays a more and more widely-used language for representing ontologies. The number of OWL ontologies increasing in direct ratio to the development of the Semantic Web leads to the heterogeneity problem. The same concepts may be modeled differently, using different terms and different positions in concept hierarchy. The task of identifying similar entities (concepts, relations or individuals) in different ontologies becomes then crucial for the success of information integration systems, instance transformation… In this paper, we propose a new similarity measure for comparing entities in different OWL DL ontologies. This measure is designed so as to enable extraction of information encoded in OWL entity descriptions and to take into account the underlying meaning of OWL primitives. We propose a variable weighting scheme for combining more efficiently component similarities calculated from components in entity descriptions.

Introduction

Several researchers currently study thoroughly problems of comparison, alignment, matching, and integration of ontologies (Euzenat, 2004). The success of these tasks depends on the way how the similarity between entities of ontologies is defined. A good measure for the similarity between two entities in two ontologies will help to identify corresponding entities correctly.

In this article, we propose a new similarity measure for any two entities in two different OWL DL ontologies. This measure is based on the information extracted from the entity descriptions (definitions). An entity in OWL DL ontology can be a class, a relation, an instance or even the ontology itself. In our system, we try to extract as much as possible information encoded in the entity description. Extracted components are compared to produce partial component similarity values. They are then combined using prefixed weights under a variable weighting scheme (where weights can be changed during the calculation, depending on situation, for the better result). The similarity calculation takes into account the predefined meanings of OWL DL and RDF(S) primitives such as *rdf:type, owl:equivalentClass*… which we can extract from entity descriptions.

A good similarity measure will be crucial for the success of several other emerging tasks in the context of semantic web such as comparing, mapping, aligning, merging or integrating ontologies as well as information.

Entity Similarity Measure

In our system, an entity in an OWL DL ontology can be a class which can have a name (an URI) or not (anonymous class), or be a relation. An entity is described in OWL DL ontology using RDF(S) or OWL DL primitives, such as *rdf:id, rdfs:range, owl:subClassOf*… Each of these primitives brings a piece of knowledge to the whole meaning of the entity. So, we can consider that the similarity between two entities in two ontologies is a combination of partial similarities which are similarities between pieces of descriptions using these primitives. The similarity combination is a variable-weighted sum calculated from partial similarities. An OWL DL ontology is an RDF document. We consider an OWL DL ontology as a set of RDF triples. Let O an ontology, let (s, p, o) a triple, where s, p, o are respectively subject, predicate, and object, O = { (s, p, o) }. Note that in OWL DL ontology, for every triple (s, p, o) $\subset$ O in entity descriptions, p is a predicate which is one of 33 RDF(S) and OWL primitives. Representation of OWL DL ontology in our system is lightly-modified RDF graph representation (Fig. 1). The labels on arcs starting from entities (classes, relations) are primitives in 33 RDF(S) or OWL primitives. For nodes which are instances, labels on arcs starting from them can be user-defined properties.

Let (s, p, o) a triple. We define:

T(e) = { (e, p, o) | (e, p, o) $\in$ O }*, the set of RDF triples having the entity e as their subject.* P(e) = { p | $\exists$o, (e, p, o) $\in$ T(e)}*, the set of predicates, which are parts of triples having entity e as their subject.* O(e, p) = { o | (e, p, o) $\in$ T(e) }*, the set of (RDF) objects, which are parts of triples having entity e as their subject and p as their predicate.* E(e) = {(p,o) | (e, p, o) $\in$ T(e) }*, the set of predicate-object pairs, where the first is predicate and the second object in a triple having entity e as subject.*

The similarity measure between two entities e_1 in ontology O_1 and e_2 in ontology O_2, named Sim(e_1,e_2) is based on two values: (1) similarity between their components and (2) similarity of their graph structure.

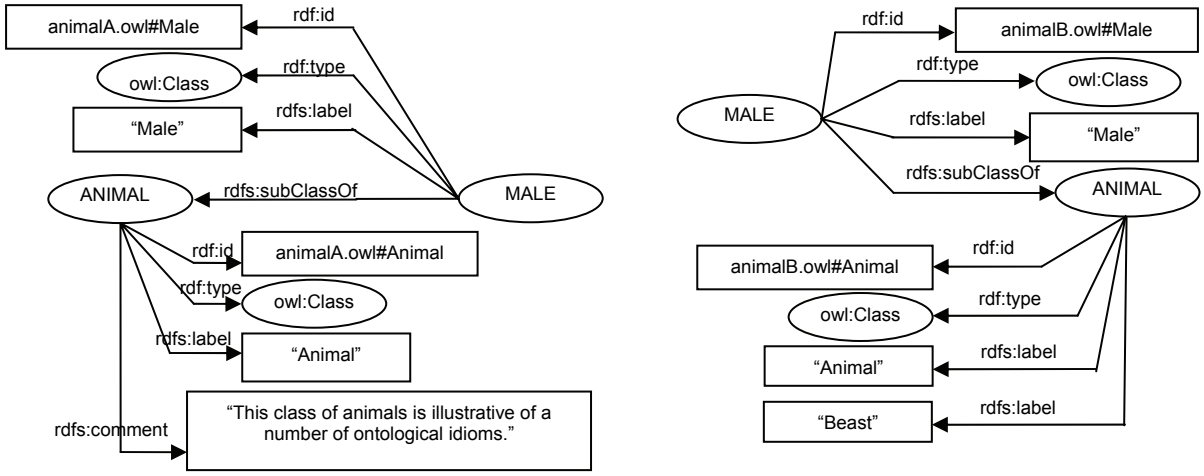


Fig. 1. Extract of modified RDF graph representations for two different ontologies: animalA.owl and animalB.owl

Similarity between Components

The components in an entity description are triples. Similarity between components of two entities is similarity between two sets of pairs $E(e_1)$ and $E(e_2)$. For comparing these two sets of pairs in order to produce a similarity value between [0,1], we propose the following steps:

1. Identify set of predicates P_{c1} which contain predicates in pairs in $E(e_1)$ having a similar predicate in a pair in $E(e_2)$ and similarly, P_{c2}. Let P_c the union of these two sets.

$P_{c1} = \{ p \mid p \in P(s_1) \wedge (\exists q \in P(s_2), SimPred(p,q) > 0) \}$

$P_{c2} = \{ p \mid p \in P(s_2) \wedge (\exists q \in P(s_1), SimPred(p,q) > 0) \}$

$P_c = P_{c1} \cup P_{c2}$

where SimPred(p,q) is a similarity function between two predicates, which are RDF(S) or OWL properties. SimPred is defined as follows: (i) SimPred(p,p) = 1; (ii) SimPred(p,q) is a predefined value in [0,1] if $p \neq q$. Some OWL properties can be considered as similar semantically, e.g. *owl:cardinality* and *owl:maxCardinality*.

2. To be able to calculate partial similarities over predicates which can appear several times in description of an entity, such as *rdfs:label, rdfs:subPropertyOf…*, we firstly collect objects in triples having this same predicate p. We thus obtain two sets of objects for two entities in two ontologies: $O(s_1,p)$ and $O(s_2,p)$. Secondly, objects in these two sets are paired, in order to calculate the similarity between object sets. The object pairing is done by the following algorithm: Let o_{1i} and o_{2j} two objects in these sets, $o_{1i} \subset O(s_1,p)$, $o_{2j} \subset O(s_2,p)$, (a) Find o_{1i} and o_{2j} whose $Sim_{total}(o_{1i},o_{2j})$ is maximal. (o_{1i},o_{2j}) is an object pair; (b) Remove o_{1i} from the set $O(s_1,p)$ and remove o_{2j} from the set $O(s_2,p)$; (c) Repeat step (a) until no more (o_{1i},o_{2j}) is found.

Finally, after the pairing process, similarities between paired objects are summed up, and then the summed value is divided by the maximal cardinality of both object sets. The obtained value is the partial similarity $Sim_{partial}$ over the considered predicate.

$$Sim_{partial}(s_1,s_2,p) = \frac{\sum_{(o_1,o_2)\in Pairing(O(s_1,p),O(s_2,p))} Sim_{total}(o_1,o_2)}{max(|O(s_1,p)|,|O(s_2,p)|)}$$

Using example in Fig. 1, $O(ANIMAL_1, rdfs:label)$ will result {"Animal"}, $O(ANIMAL_2, rdfs:label)$ will give {"Animal", "Beast"}. Pairing two sets gives {("Animal", "Animal")}. So the $Sim_{partial}$ over predicate *rdfs:label* for two classes ANIMAL will be 0.5.

Some predicates, such as *owl:equivalentProperty, owl: sameAs, owl:equivalentClass* which can appear several times in the description of an entity, require a different processing. Instead of pairing objects in triples having the same considered predicate in order to compute the partial similarity from the similarity between two sets of objects, the partial similarity is calculated based on the meaning of the predicate, and in this case, is the maximal similarity value of objects in two sets.

$$Sim_{partial}(s_1,s_2,p) = \max_{o_1\in O(s_1,p),o_2\in O(s_2,p)} (Sim_{total}(o_1,o_2), Sim_{total}(s_1,o_2), Sim_{total}(o_1,s_2))$$

3. For predicates which can only appear at most once in any entity description, such as *rdf:id, owl:complementOf, owl:inverseOf…*, the partial similarity is only based on two objects and the underlying meaning of the predicate. Function θ, depending on meaning of given predicate and objects, returns similarity value of two entities.

$$Sim_{partial}(s_1,s_2,p) = \theta(p, Sim_{total}(o_1,o_2))$$

As example, if the predicate is *owl:complementOf*, the partial similarity of two entities is based on the similarity between two objects of two triples: if two objects are similar classes, their complementary classes are also similar. The $Sim_{partial}$ of two entities over the predicate *owl: complementOf* will then be the Sim_{total} of the two objects.

4. Depending on the ontology modeling, description of an entity (class, relation) can consist of one or several RDF(S) /OWL properties. A given property (predicate) can appear in the description, others not. The total similarity between two entities is a combination of their partial similarities calculated as described above: it is a weighted sum from the obtained partial similarities. As discussed, the entity

description components are not fixed, they vary from entity to entity. So, applying a fixed weighting scheme over partial similarities might be not efficient. Let us use a simple example: suppose that we have a simple ontology language which supports only three property primitives: *ol:id, ol:label* and *ol:comment*. An entity description is an arbitrary combination of these primitives: one can be defined using only *ol:id* for its identification, another one can be described by all three primitives for its identification, its human-readable name and comment. For two entities, our method will produce partial similarities from these components. In related research for combining partial similarities, each component is assigned a pre-fixed weight (e.g. 0.5, 0.35 and 0.15 respectively), and then the final result is the weighted sum of products of partial similarities and these weights. If the descriptions for both entities being compared contain only one primitive *ol:id*, the maximal similarity of two entities is only 0.5 (in the case that both identifications are same). To solve this problem, we propose a variable weighting scheme where predefined weights can be modified automatically in the calculation depending on descriptions of entities being compared. Using previous example, notifying that both entities have only one primitive *ol:id* in their descriptions, the corresponding weight for this primitive changes from 0.5 to 1.0, thus the obtained total similarity value may reach the value of 1.0.

$$Sim_{component}(s_1, s_2) = \sum_{p_i \in P_c} \phi(w_{p_i}) * Sim_{partial}(s_1, s_2, p_i)$$

$\phi(w)$ is an adaptation function for modifying weights.

There are several weight-changing strategies. We propose the following one: (1) Initiate the 33 pre-fixed weights corresponding to the 33 RDF(S)/OWL primitives, so that their sum is equal to 1.0. These weights are firstly assigned manually to different values to give different emphasis to different components (corresponding to primitives) in entity descriptions. The fact that for entity similarity calculation, the information getting from *rdf:id* is more important than from *owl:versionInfo*, so the weight assigned to the former is set higher than one assigned to the latter; (2) When comparing two entities, for each RDF(S)/OWL primitive p_i which does not appear in the descriptions of both entities, set (automatically) its corresponding weight from w_i to 0.0 and increase (automatically) all other non-zero weights by an amount being equal to w_i/[number of non-zero weights]. This guarantees that their sum is always equal to 1.0.

Similarity of Entity Graph Structures

An entity description is represented by an RDF graph. Similarity between two entities can be derived not only from similarities between their description components, but also from the similarity between the structures of the RDF graphs representing them. In our system, the entity graph similarity is formulated from the ratio between the number of similar predicates over the maximal number of triples of both entities. Taking into account the underlying meaning of RDF(S)/OWL properties, some properties are considered as similar, e.g. *owl:cardinality, owl:maxCardinality* and *owl:minCardinality*. The formulation does not regard the similarity of objects, but only the similarity of predicates of two entities.

$$Sim_{graph}(s_1, s_2) = \frac{|P_c|}{max(|P(s_1)|, |P(s_2)|)}$$

Total Similarity

The total similarity between two entities is the combination of two similarity values: their component similarity $Sim_{component}$ and their graph structure similarity Sim_{graph}. Here, a fixed weighting scheme is applied for the combination. The reason is that these two similarity components are not optional, they exist for every pair of entities. The weights can be chosen following experimental results.

$$Sim_{total} = w_{component} * Sim_{component} + w_{graph} * Sim_{graph}$$

where $w_{component} + w_{graph} = 1$

Implementation and Results

Our entity similarity measure was implemented in Java. Jena API [1] was used for loading OWL DL ontologies into memory. The OWL DL graph representation is derived from the Jena RDF model.

Ontologies used for evaluation are ontologies proposed in the context of the I^{3}CON conference[2] even though we did not compete in 2004. To evaluate our similarity measure, a very simple algorithm is installed: for each entity (class, property) in an ontology, the similarity value with all other entities in the other ontology is calculated using our measure. The maximal similarity value will be shown out. Note that our goal is to test the measure, not the efficiency of the matching algorithm.

Order	Entity in ontology animalA.owl	Entity in ontology animalB.owl	Similarity value
1	Shoesize	shoesize	1.0
2	Shirtsize	Shirtsize	1.0
3	Animal	Animal	1.0
…			
23	hasFemaleParent	hasWife	0.037
24	hasMaleParent	hasHusband	0.034

Table 1: compare ontology animalA.owl with animalB.owl

We calculated similarity values (see table 1) between entities in ontology *animalA.owl* (having 35 entities) and entities in ontology *animalB.owl* (having 24 entities). The latter is a modified version of the former. It has no instance and it has reduced entity descriptions. Note that this simple non-optimized algorithm is not recursive and has no similarity matrix. Its total averaged running time for the test is about 2.204 second. As result, we have 3 entity pairs with similarity value of 1.0. The worst value is 0.034 for the (incorrect) pair *hasMaleParent-hasHusband*. All we know about the entity *hasMaleParent* is information extracted from its description. This informs us that the entity type is *owl:ObjectProperty* and that it is *owl:equivalentProperty* with *hasFather*. As the algorithm

[1] http://jena.sourceforge.net/

[2] http://www.atl.external.lmco.com/projects/ontology/i3con.html

does not store temporary similarity values, it does not know that *hasFather* in *animalA.owl* ontology is exactly similar (similarity value of 1.0) to the entity *hasFather* in *animalB.owl* ontology. That explains why *hasMaleParent* is not matched with *hasParent* but *hasHusband* in the second ontology. But in the future work, we will adapt an incremental algorithm which can overcome these limits.

Related Work

The closest related work with our proposition is the work of (Euzenat and Valtchev, 2004). Similarity between two nodes depends on categories which they belong to and relations between categories. It is a fixed weighted sum of partial similarities while we apply a variable weighting scheme. (Weinstein & Birmingham, 1999) proposes several similarity measures for comparing concepts in differentiated ontologies. Their measures are mainly based on the compatibility comparison of structural descriptions and do not rely on the underlying meaning of relations between concepts. (Maedche & Staab, 2002) presents another work for measuring the global similarity between the whole two ontologies. Contrarily to theirs, our measure is for the similarity between entities in two ontologies.
Other researchers (Doan *et al*., 2002), (Melnik *et al*., 2002), (Noy and Musen, 2000) propose algorithms for finding entity mappings between two schemas (simple ontologies). They do not focus specially on constructing a similarity measure as we do. For lack of room, we don't detail all related work but good surveys can be found in (Rahm and Bernstein, 2001), in (Kalfoglou and Schorlemmer, 2003) and in (Euzenat, 2004).

Conclusions and Future Work

In this paper, we presented our new measure for calculating similarity of two entities (classes, relations) from two different OWL DL ontologies. These ontologies are represented in RDF-like graphs and the similarity measure is formulated based on two parts: (1) similarity between the components of the entity descriptions and (2) similarity between graphs representing entities. For the first part, we proposed methods for dealing with the similarity of two sets and for combining component partial similarities in a variable weighting scheme. Our similarity measure also takes into account the underlying meanings of RDF(S) and OWL properties, which are used in entity descriptions. The measure was implemented in Java and tested with entities in OWL ontologies for the validation. In addition to I^{3}CON[1], we will test our measure on EON[2] and on two real-world ontologies O'COMMA and O'Aprobatiom compared with our previous algorithm (Bach *et al*, 2004).

Many algorithms based on similarity measures to discover mappings can be developed using our measure. A mapping will be created for two entities in two different ontologies when these entities are considered as (semantically) similar, i.e. when their similarity value is higher a certain threshold. For future work, we will focus on the integration of our proposed measure in an ontology matching algorithm or in a merging algorithm and evaluate its performance and efficiency in these real and crucial tasks.

References

Bach, T.L., Dieng-Kuntz, R., Gandon, F. 2004: On *Ontology Matching Problems - for Building a Corporate Semantic Web in a Multi-Communities Organization*. In Proc. of ICEIS'04, Porto, Portugal, 2004, p. 236-243.

Cohen, W. W., Ravikumar, P., and Fienberg, S. 2003. *A Comparison of String Distance Metrics for Name-Matching Tasks*. IJCAI'2003 Workshop on Information Integration on the Web.

Deborah, L. M., Fikes, R., Rice, J., and Wilder, S. 2000. *An Environment for Merging and Testing Large Ontologies*. In Proc. of KR'00. Breckenridge, Colorado, USA. April 12-15, 2000.

Dieng, R. and Hug, S. 1998. *Comparison of "Personal Ontologies" Represented through Conceptual Graphs*. In Proc. of ECAI'98, p. 341-345, Brighton, UK.

Doan, A., Madhavan, J., Domingos, P., and Halevy, A. 2002. *Learning to Map between Ontologies on the Semantic Web*. In Proc. of WWW'02, Hawaii, USA.

Euzenat, J., Valtchev, P.: *Similarity-based ontology alignment in OWL-Lite*. ECAI'04, Valencia, pp333-337

Euzenat, J.: *State of the art on current alignment techniques*, 12/2004, IST Knowledge Web Network of Excellence no FP6-507482, #KWEB/2004/ D2.2.3.

Kalfoglou, Y. and Schorlemmer, M. 2003. *Ontology mapping: the state of the art*. The Knowledge Engineering Review Journal, (18(1)):1-31, 2003.

Maedche, A. and Staab, S. 2002: *Measuring Similarity between Ontologies*. In Proc. of EKAW'02. Madrid, ES.

Melnik, S., Garcia-Molina, H., and Rahm, E. 2002. *Similarity Flooding: A Versatile Graph Matching Algorithm and its Application to Schema Matching*. In Proc. 18th ICDE'02, San Jose CA.

Noy, N. F. and Musen, M. A. 2000. *PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment*. In Proc. of AAAI'00, Austin, TX.

Rahm, E. and Bernstein, P. A. 2001. *A survey of approaches to automatic schema matching*. In The VLDB Journal: Volume 10 Issue, pages 334-350.

Weinstein, P., and Birmingham, W.: *Comparing concepts in differentiated ontologies*. In Proc. of KAW'99, 1999.

[1] http://www.atl.external.lmco.com/projects/ontology/i3con.html
[2] http://co4.inrialpes.fr/align/Contest/

Enhance Reuse of Standard e-Business XML Schema Documents

Buhwan Jeong[1], Boonserm (Serm) Kulvatunyou[2]*, Nenad Ivezic[2], Hyunbo Cho[1], Albert Jones[2]

1Pohang University of Science and Technology, San 31, Hyoja, Pohang, 790-784, South Korea

2National Institute of Standards and Technology, 100 Bureau Drive, Gaithersburg, MD 20899, USA, *Corresponding Author

Abstract

Ideally, e-Business application interfaces would be built from highly reusable specifications of business document standards. Since many of these specifications are poorly understood, users often create new ones or customize existing ones every time a new integration problem arises. Consequently, even though there is a potential for reuse, the lack of a component discovery tool means that the cost of reuse is still prohibitively high. In this paper, we explore the potential of using similarity metrics to discover standard XML Schema documents. Our goal is to enhance reuse of XML Schema document/component standards in new integration contexts through the discovery process. We are motivated by the increasing access to the application interface specifications expressed in the form of XML Schema. These specifications are created to facilitate business documents exchange among software applications. Reuse can reduce both the proliferation of standards and the interoperability costs. To demonstrate these potential benefits, we propose and position our research based on an experimental scenario and a novel evaluation approach to qualify alternative similarity metrics on schema discovery. The edge equality in the evaluation method provides a conservative quality measure. We review a number of fundamental approaches to developing similarity metrics, and we organize these metrics into lexical, structural, and logical categories. For each of the metrics, we discuss its relevance and potential issues in its application to the XML Schema discovery task. We conclude that each of the similarity measures has its own strengths and weaknesses and each is expected to yield different results in different search situations. It is important, in the context of an application of these measures to e-Business standards that a schema discovery engine capable of assigning appropriate weights to different similarity measures be used when the search conditions change. This is a subject of our future experimental work.

An Experimental Scenario and Evaluation[.]

Experimental Scenario

We propose a Schema Discovery Engine that applies different combinations of similarity metrics to one or more relevant, standard, document (component) schemas that may satisfy given integration requirements. Figure 1 illustrates the experimental evaluation planned for our Schema Discovery Engine running a similarity metric.

We will use test data from a real, industrial integration problem involving B2B data exchange. The component library, stored in the repository on the left-hand side of the figure, will be based on the Open Applications Group Integration Specification (OAGIS) [3]. The OAGIS is a horizontal standard for business data exchange including supply chain data. Sample data exchange requirements (originally captured in a class model or SQL statements) will be taken from the Automotive Industry Action Group (AIAG) supply chain standards and the Oracle ERP interfaces as shown on the right-hand side of the figure. To facilitate our computing environment, the data exchange requirements will be translated into a common syntax such as an XML Schema or a pseudo XML instance. Those requirements have maps, with possible extensions, to the OAGIS. The maps are considered to be correct and will be compared against discovered components in the evaluation phase.

Figure 1 shows how this might work from the requirement in the AIAG Min/Max Vendor Managed Inventory scenario called the *QuantityOnHand* (QOH) [4]. The *QOH* model indicates that the required data fields are *Item*, *SiteId*, *Quantity*, *MinQuantity*, and *MaxQuantity*. A user who has the model of this component searches the library for reusable components. The schema discovery engine uses the QOH model information and any other relevant documentation to calculate similarities against the components in the library. It can evaluation options that consists of several combinations of different types of similarity measures to determine the best potential matches. The user can choose different combination options, such as the Harmonic mean, and set the threshold that determines how many and which kinds of components are returned.

In the figure, the *InventoryBalance* and the *WIP* components whose overall similarity values are above the threshold, 0.7, are returned. Within each final result, individual similarities are computed indicating the strength of the mapping between each field within the discovered component and each field in the requirement. Within the illustration, the discovery engine might not be able to identify any fields with sufficiently high similarity measures to induce equivalences for the *MinQuantity* and the *MaxQuantity* fields; however, it could indicate that the two fields could establish some relationships with the *Quantity* field. The relationships may include equivalent, more (or less) general, and overlapping [1].

We expect that the results from such an analysis could guide users by making better and more efficient judgments about the potential reuse of existing schemas. In Figure 1,

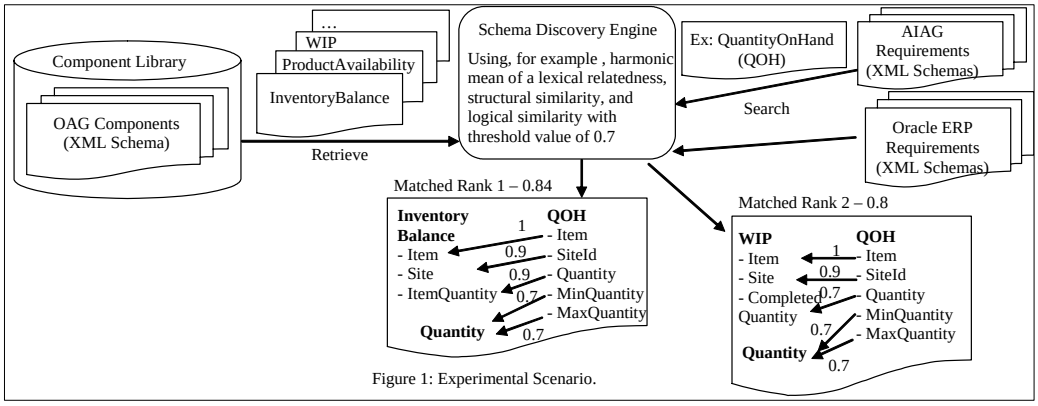


Figure 1: Experimental Scenario.

the result could be interpreted to mean that the *QOH* may be designed appropriately as an extension of the *InventoryBalance* components where the *MinQuantity* and *MaxQuantity* are the extensions of the basic *Quantity* field. Additionally, the discovery result also points to the *WIP* component as a possible basis for the *QOH* component. It is important to note that extensions to existing components should be added to the component library so that they could be discovered and reused in subsequent integration activities.

Experimental Evaluation

In the previous section, we said that the schema discovery engine selectively uses and combines similarity metrics. In this section, we illustrate an example approach to evaluate and compare the schema discovery quality in different combinations of similarity measures.

Since the components in the library and the requirements are represented using an XML tree-based structure, we argue that each data field, either element or attribute, can be addressed using the XPATH expression [2]. In Figure 1, we can address the fields as *InventoryBalance*/*Item* or *QOH*/*Item* and *InventoryBalance*/*Site* or *QOH*/*SiteId*, for example.

Let a set $U = \{u_i\}$, $i = 1, 2, ..., n$ be a set of XPATH expressions, u_i, for each element or attribute field of the target component (the requirement) U. Similarly, let a set $V = \{v_j\}$, $j = 1, 2, \ldots, m$ be that of the true mapped component(s) from the library, and a set $W = \{w_k\}$, $k = 1, 2, ..., p$ be that of a discovered component.

Then, let a set of edge constraint $E_t = \{ e_t = (u'_{i'}, v'_{j'}) \}$, $i' = 1, 2, ...,n'$ and $j' = 1, 2, ..., m'$, $n' \le n$ and $m' \le m$ be the true map from the fields in U to V. Similar to E_t, let $E_d = \{ e_d = (u'_{r'}, w'_{k'}) \}$, $r' = 1, 2, ..., r$, $k' = 1, 2, ..., p'$, $r \le n$, and $p' \le p$ be the map from U to W as estimated by the discovery engine. A graphical representation of E_d is as shown in the output from the schema discovery engine in Figure 1. As shown in the figure, there may be some fields in U that have no map to any field in W (e.g., the *MinQuantity* and the *MaxQuantity* fields). Alternatively, components may be discovered for which some of the fields cannot be mapped (used) onto the required fields. It should be noted that in practice, $u'_{i'}$ ($w'_{k'}$) can be either a u_i (w_k) or a composition of two or more u_i (w_k). Although not shown, a graphical representation of E_t and relationship between $v'_{j'}$ and v_j are similar to that of E_d. Using the above definitions and typical information retrieval measurements, we can measure the quality of a discovered component d,

using *Jaccard* as $Q_d = | E_t \cap E_d | / | E_t \cup E_d |$ or
using *Recall* as $Q_d = | E_t \cap E_d | / | E_t |$ or
using *Precision* as $Q_d = | E_t \cap E_d | / | E_d |$

where two edges $e_t \in E_t$, and $e_d \in E_d$ are matched, i.e., $e_t = e_d$ if and only if the paths $u'_{i'} = u'_{r'}$ and $v'_{j'} = w'_{k'}$.

If the schema discovery engine returns multiple components, an example of the overall discovery quality could be $Q_D = max_{d \in D} (Q_d)$, where D is a set of discovered components.

There are some issues with this discovery quality measure. First, the edge equality definition makes this quality measure a conservative one, because it requires that the labels on both paths be identical. For a discovered component where some labels are different yet semantically equal, the quality would be unrealistically low.

Second, the discovery quality (Q_D) may not indicate the performance of the overall system, if our intent is to consider multiple alternatives. In that case, the measure has to be normalized against the number of suggestions returned. In addition, the discovered component providing the maximum quality, *max* (Q_d), may not be the component ranked the most similar by the discovery engine. Thus, we want the discovery engine to produce a similarity value that is highly correlated with the quality associated with the component. We envision the correlation value between the similarity value of a component d and Q_d across the members of the set of discovered components D as a dimension of the overall discovery quality. The advantage of such quality measure is that it is orthogonal to the number of discovered components.

Similarity Metrics: A Literature Review

We organize the review of similarity approaches into three groups: lexical, structural, and logical. For each of the approaches, we discuss its relevance and potential issues in applying it to the XML Schema discovery task. In closing this section, we give our perspective on the respective roles and potentials of the investigated categories of

similarity metrics, both considered individually and in combination.

Lexical Perspective

A lexical similarity measure quantifies the commonality between individual component names using purely lexical information. Commonly used lexical similarity measures include *Tanimoto* [19], n-gram [18], (weighted-) distance-based [5, 6, 13, 14], word sense-based [1], and information content-based [7] metrics.

We found that the existing lexical similarity measures may not be directly applicable to our schema discovery problem. The reason is that an XML component name usually consists of several words and/or allowable abbreviations concatenated to enhance their expressivity. Such composite words (e.g., *QuantityOnHand, InventoryBalance*) provide more information than individual words because the additional words provide additional context information. Moreover, the composite words make the meaning of the included words more specific. This information is particularly important when a domain-specific lexical resource is not available. For example, we can eliminate several senses associated with the term *Contact* within the component name *DeliveryToContact*. Because *Contact* follows the verb, it must be a noun. Further, the *relationship* and the *surface* senses of the *Contact* can be eliminated because one would not deliver a product to a relationship or a surface in the business sense. Hence, the similarity measure should be constructed to focus on the meaning of the *Contact* associated with a person. Furthermore, we envision that each word in a component name should have different salience depending on its part of speech. For example, we would like the component name *DeliveryToContact* to have a higher similarity value when comparing it to the *ShipToContact* than to the *DeliveryFromContact* since the latter is, in fact, an opposite. The research to advance the lexical similarity measures for the schema discovery should exploit this type of additional information.

We also recognize that domain-specific resources are very important in analyzing lexical similarity. Consequently, our future research may include methods to model domain-specific resources in our supply-chain and logistics problem contexts. In addition, the schemas and requirements documentations are context specific resources for the content-based similarity analysis.

Structural Perspective

A structural similarity measure quantifies the commonality between components by taking into account the lexical similarities of multiple, structurally related sub-components of these terms (e.g., child components, child attributes). A structural similarity metric typically provides a more conservative measure than the lexical similarity, because it looks beyond the individual labels and their definitions to the context surrounding these labels. The tree structure is a native structure for XML documents; hence, it is most related to our problem context. While significant research has been done to apply these methods to XML instance documents, they may be applied to schema discovery by representing the XML schema using one or more pseudo XML instances. Commonly used structural similarity measures include node, path, and/or edge matching, tree edit distance (TED) [8, 9, 12], (weighted) tag similarity [9], weighted tree similarity [10], and Fourier transformation-based approach [15].

Although existing structural similarity measures can be useful in schema discovery, there are several issues that need to be addressed. First, the existing measures are geared toward content rather than meta-data; hence, the perspective of these approaches needs to be adjusted.

Second, one of the most powerful structural measures, TED, is more applicable to ordered trees because this insures computability in polynomial time. However, the order constraint does not always apply to schemas; hence, further research is required to determine conditions under which this restriction can be relaxed. One possible approach is to re-order and represent schemas in abstract tree structures. Another is to ignore the structure in local areas and aggregate them into a single node. The less powerful measures such as path or inclusive path matching do not exploit fully context-specific information embedded in the structural relationships. The weighted measures require a practical way to obtain weights.

Logical Perspective

A logical similarity measure quantifies the commonality of properties/constraints restraining components definitions beyond the lexical and structural aspects such as type, cardinality, etc. The logical similarity is often classified as a structural category [18, 19]. However, we treat it as an independent category because it is the most restrictive and accurate measure. That is, even if two components have identical label and structures, their logical similarity value can still be imperfect.

Take a term *TelephoneNumbe*r, which consists of two child elements: an *AreaCode* element followed by a *Number* element. Suppose that there are two *TelephoneNumber* definitions, one defines the types (ranges) associated with child elements as *Integer* while the other defines them as *String*. Although the two have exact labels and structures, a good logical similarity measure would indicate that they are not identical and potentially incompatible. The logical similarity measures can provide more powerful estimates when matching schemas using additional model-based information. For example, if there is model-based information that indicates that the *String* type subsumes the *Integer* - indicating the *Integer* is convertible to the *String*, but not vice versa- then the measure may be used to indicate that the term is always translatable to the other but not vice versa. Some example approaches in this category include DL-based [1, 17], instance-based [16], and graph-based [11] approaches.

Although the logical similarity measures are potentially more accurate due to their formal basis, they require the

model to provide significant additional information, which is often unavailable. When model-based information is shallow, the quality of the approach may be reduced drastically. Hence, any schema discovery engine using logical similarity measures has to adjust the weights based on the amount and kind of model-based information available. In particular, a lower weight should be given to the logical similarity if the subsumption hierarchy is very shallow.

Finally, we offer our synthesized view of the respective roles and potentials uses of the aforementioned similarity metrics on the XML Schema discovery task. Schema discovery in the enterprise-applications-integration context is a unique information retrieval problem, because the goal is not to retrieve the content but the data model associated with the content. Specific consideration must be given to terms and naming conventions, design and structure conventions, usage cases, and semantic/ontology models, all of which must be considered simultaneously when matching schemas to a requirement. Therefore, it is not likely that a single similarity category would yield optimal results.

Synthesis of various similarity metrics within a search algorithm is likely to produce more accurate results. However, achieving such a synthesis is not straightforward. On the one hand, lexical measures may be more effective when a domain-specific thesaurus or dictionary is available. On the other hand, structural measures will be more effective when the data exchange requirements and the standard specification schemas within the repository are similarly constructed or are known to follow the same design conventions. In such well-controlled situations, the two similarity metric categories may play more deterministic roles, while the measures within the logical similarity category may appropriately play an auxiliary role, particularly when the schemas and the requirement are totally disparate.

References

1. Giunchiglia, F., Shvaiko, P., and Yatskevich. M. 2004. S-Match: An Algorithm and an Implementation of Semantic Matching. In *Proc. of ESWS*:61-75.
2. WWW Consortium. 1999. *XML PATH Language 1.0*.
3. The Open Application Group. 2002. *Open Application Group Integration Specification version 8.0*.
4. Automotive Industry Action Group. 2005. *Proof of Concept Phase 1 Project Summary*.
5. McHale, M. 1998. A Comparison of WordNet and Roget's Taxonomy for Measuring Semantic Similarity. In *Proc. of the COLING/ACL Workshop on Usage of WordNet in Natural Language Processing Systems*, Montreal, Canada:115-120.
6. Jarmasz, M., and Szpakowicz, S. 2003. Roget's Thesaurus and Semantic Similarity. In *Proc. of Conf. on Recent Advances in Natural Language Processing (RANLP)*, Borovets, Bulgaria:212-219.
7. Resnik, P. 1995. Using Information Content to Evaluate Semantic Similarity in a Taxonomy. In *Proc. of the 14th Intl. Joint Conf. on AI*, Montreal, Canada:448-453.
8. Zhang, Z., Li, R., Cao, S., and Zhu, Y. 2003. Similarity Metric for XML Documents. In *Proc. of Workshop on Knowledge and Experience Management*, Karlsruhe, Germany.
9. Buttler, D. 2004. A Short Survey of Document Structure Similarity Algorithms. In *Proc. of the 5th Intl. Conf. on Internet Computing*, Las Vegas, Nevada.
10. Bhavsar, V.C., Boley, H., and Yang, L. 2003. A Weighted-Tree Similarity Algorithm for Multi-Agent Systems in e-Business Environments, In *Proc. of the Business Agents and the Semantic Web (BASeWEB)* Workshop, Halifax, Nova Scotia, Canada.
11. Noy, N.F., and Musen, M.A. 2001. Anchor-PROMPT: Using Non-Local Context for Semantic Matching. In *Proc. of the Workshop on Ontologies and Information Sharing at the 17th Intl. Joint Conf. on Artificial Intelligence*, Seattle, WA.
12. Nierman, A. and H.V. 2002. Evaluating Structural Similarity in XML Documents. In *Proc. of the 5th Intl. Workshop on the Web and Databases*, Madison, WI.
13. Sussna, M. 1993. Word Sense Disambiguation for Free-Text Indexing using a Massive Semantic Network. In *Proc. of the 2nd Intl. Conf. on Information and Knowledge Management*, Arlington, VA.
14. Richardson, R., and Smeaton, A.F. 1995. *Using WordNet in a Knowledge-based Approach to Information Retrieval*, Working Paper, School of computer applications, Dublin City University, Ireland.
15. Flesca, S., Manco, G., Masciari, E., Pntieri, L., and Pugliese, A. 2002. Detecting Structural Similarities between XML Documents. In *Proc. of the 5th Intl. Workshop on the Web and Databases*, Madison, WI.
16. Doan, A., Madhavan, J., Domingos, P., and Halevy, A. 2003. Learning to Match Ontologies on the Semantic Web, *VLDB Journal, Special Issue on the Semantic Web*.
17. Peng, Y., Zou, Y., Luan, X., Ivezic, N., Gruninger, M., and Jones, A. 2003. Semantic Resolution for e-Commerce, Innovative Concepts for Agent-Based Systems. *Springer-Verlag* :355-366.
18. Do, H. and Rahm, E. 2001. COMA: A System for Flexible Combination of Schema Matching Approaches, In *Proc. of VLDB*, Roma, Italy:610-621.
19. Duda, R., Hart, P., and Stork, D. 2001. Pattern Classification, 2nd Edition, *Wiley-Interscience*.
20. Castano, S., De Antonellis, V., and De Capitani di Vimercati, S. 2001. Global Viewing of Heterogeneous Data Sources, *IEEE Transactions on Knowledge and Data Engineering* 13(2):277-297.

Disclaimer

Certain commercial software products are identified in this paper. These products were used only for demonstration purposes. This use does not imply approval or endorsement by NIST, nor does it imply these products are necessarily the best available for the purpose.

What Is Ontology Merging?

– A Category-Theoretical Perspective Using Pushouts

Pascal Hitzler and **Markus Krötzsch** and **Marc Ehrig** and **York Sure**

Institute AIFB, University of Karlsruhe, Germany;
{hitzler,kroetzsch,ehrig,sure}@aifb.uni-karlsruhe.de

Introduction

In this paper we explain how merging of ontologies is captured by the pushout construction from category theory, and argue that this is a very natural approach to the problem. We study this independent of a specific choice of ontology representation language, and thus provide a sort of *blueprint* for the development of algorithms applicable in practice. For this purpose, we view category theory as a universal "meta specification language" that enables us to specify properties of ontological relationships and constructions in a way that does not depend on any particular implementation. This can be achieved since the basic objects of study in category theory are the *relationships* between multiple ontological specifications, not the internal structure of a single knowledge representation.

Categorical pushouts are already considered in some approaches to ontology research (Jannink *et al.* 1998; Schorlemmer, Potter, & Robertson 2002; Goguen 2005; Kent 2005) and we do not claim our treatment to be entirely original. Still we have the impression that the potential of category theoretic approaches is by far not exhausted in todays ontology research. For our particular case the treatment will focus on the ontology merging, for which we will give both intuitive explanations and precise definitions. This reflects our belief that, at the current stage of research, it is not desirable to fade out the mathematical details of the categorical approach completely, since the interfaces to current techniques in ontology research are not yet available to their full extent. We will also keep this treatment rather general, not narrowing the discussion to specific formalisms – this added generality is one of the strengths of category theory.

A long version of this paper with a tutorial character is available from the first author's homepage.

Categorical preliminaries

In order to approach the concept of a category, we view it as a system of ontological specifications that includes both ontologies and their interrelations. Informally, an ontology can be viewed as something which conveys a certain specification (e.g. of some data) based on a given classification system. Mathematically, this description allows for a number of realizations: tree structures, formal contexts, partially ordered sets, or deductive systems of some logic are only examples. These approaches vary widely in their expressive power and may appear rather diverse indeed.

On the other hand, any suitable notion of an ontology should feature certain properties. This derives from the fact that ontologies are conceived as a means of sharing and reusing knowledge. Hence a typical task is to compare several (specifications of) ontologies or to combine them into a more extensive one. The latter process is often termed *ontology merging*, and will be discussed herein. For the sake of simplicity, our examples will use is-a hierarchies in order to explain the abstract notions involved.

The mathematical structure of is-a hierarchies is simply that of partially ordered sets, posets in short. Two posets can be considered to be *equivalent*, if there exists a bijective function (i.e. one which is one-to-one and onto) between these sets which does also preserve the order (i.e. which is *monotonic*). In this case, being monotonic means that a function respects the internal structure of partially ordered sets, while bijectivity indicates the equivalence of two ordered sets. Structure-preserving functions are a typical implementation of what is called a *morphism* in category theory, and what we will recognize as a suitable substitute for the consideration of internal structures.

While monotonic functions are reasonable morphisms for comparing posets, other mathematical spaces may suggest different kinds of morphisms: Vector spaces are considered with linear functions, groups with group homomorphisms, topological spaces with continuous functions, etc. The idea that emerges from these observations is that the relationships between objects are basically captured by the morphisms that exist between them. By deciding for a particular type of morphisms, we determine which internal properties of the mathematical objects are considered "essential" (e.g. order structure or cardinality). This is the approach taken in category theory: a class of *objects* (e.g. order structures) is equipped with *morphisms* (e.g. monotonic functions), thus forming a large directed graph with objects as nodes and morphisms as arrows. Depending on the given situation, arrows can be identified with certain functions or relations between the entities that were chosen for objects, but no such concrete meaning is required. In order to constitute a cate-

gory, a directed graph only has to include a *composition* operation (denoted $\circ$) for pairs of compatible arrows, satisfying some simple axioms that are typical for the composition of functions and the relational product. A precise definition is found in the aforementioned full version of this paper.

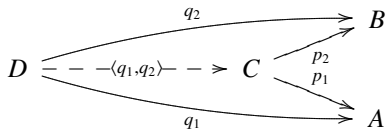
Specific relationships between objects can now be defined in purely categorical terms. Let us explore the notion of equivalence (or, speaking categorically, *isomorphism*) for the category of posets and monotone functions, where composition of morphisms is just usual composition of functions. Restating our earlier insights, we find that two partially ordered sets P and Q are equivalent (isomorphic) whenever there is a monotone function $f : P \to Q$ that has a monotone *inverse*, i.e. for which there is a monotone function $g : Q \to P$ with $g \circ f = \mathrm{id}_P$ and $f \circ g = \mathrm{id}_Q$. Generalizing this to arbitrary categories, we call a morphism an *isomorphism* if it has a (necessarily unique) inverse morphism. We continue next with discussing some constructions which will help in understanding pushouts.

Products and Relations

In set theory, the cartesian product of two sets is defined as the set of all pairs of elements from two given sets. This is not a suitable description from the viewpoint of category theory, since we want to avoid to mention the internal (element-based) structure of our objects. In order to rephrase this in categorical language, we need to find alternative criteria that rely exclusively on properties of the morphisms. To this end, an important observation is that a product does in general also provide two *projection functions* to the first respectively second component of the product. Furthermore, the product is distinguished by a *universal property* given in the next definition.

Definition 1 Consider a category $\mathscr{C}$ and objects $A, B \in |\mathscr{C}|$. Given an object $C \in |\mathscr{C}|$ and morphisms $p_1 : C \to A$ and $p_2 : C \to B$, we say that (C, p_1, p_2) is the *product* of A and B if the following universal property holds:

For any object $D \in |\mathscr{C}|$ and morphisms $q_1 : C \to A$ and $q_2 : C \to B$, there is a unique morphism $\langle q_1, q_2 \rangle : D \to C$, such that $q_1 = p_1 \circ \langle q_1, q_2 \rangle$ and $q_2 = p_2 \circ \langle q_1, q_2 \rangle$. The latter situation is depicted in the following diagram:



For example, in set theory for the usual cartesian product, we can define the function $\langle q_1, q_2 \rangle$ by setting $\langle q_1, q_2 \rangle(d) = (q_1(d), q_2(d))$. In spite of this, the above defines the cartesian product of sets only up to isomorphism (i.e. bijective correspondence) – *any* set with the cardinality of the cartesian product can be equipped with appropriate morphisms. This is a typical feature of category theory: isomorphic objects are not distinguished, since they behave similar in all practical situations. It is the choice of morphisms that determines what distinctions are considered relevant in the first place. We also remark that products do not necessarily exist in every category.

We remark, nevertheless, that the notion of *product* of two objects depends solely on the chosen category, i.e. on the objects and their morphisms. Fixing, for example, a specific ontology language, and finding an agreement on which features of an ontology should be preserved by a corresponding morphism, we obtain a notion of *product* in a canonical way.

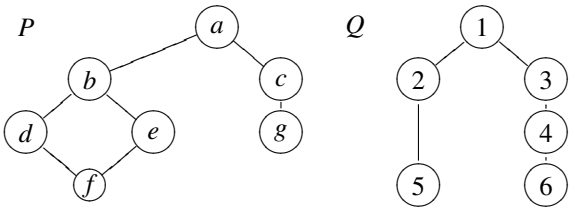
The categorical product definition also turns out to be suitable to model many well-known product constructions. For instance, when considering posets and monotone functions one obtains the usual product order, i.e. the cartesian product of the two sets, ordered such that a pair (a, b) is below a pair (c, d) whenever a is below c and b is below d.

By means of the product construction, we can also introduce *binary relations* on objects. Indeed an ordinary set-theoretic binary relation is just a subset of the cartesian product of two objects. Hence it makes sense to consider a morphism $r : D \to (A \times B)$ from some object D to the product of A and B as a binary relation between A and B. Note that this does also give us two functions $p_1 \circ r : D \to A$ and $p_2 \circ r : D \to B$ to the two components of the product, for which the morphism r is already the unique factorization that exists due to the definition of a product.

Merging ontologies via pushouts

We will now return to our initial motivation. Our intuition is that the objects of our category represent ontologies and that the morphisms between them serve as meaningful transitions between these specifications. The categorical product construction is not suitable for the purpose of modelling ontology merging, since it does obviously not consider any relationship between two ontologies. Such a relationship – commonly referred to as an *ontology mapping* – however is the base of an ontology merging process, so we have to find a means of modelling it in our categorical setting. We are in fact more interested in a certain kind of *sum* than in a product. Indeed, if two ontologies were entirely unrelated, they could be combined by just taking their disjoint union (provided that this operation makes sense for the chosen ontology representation language). However, we are more interested in merging ontologies that do overlap (via some mapping), where some elements are related while others are not. Merging two such ontologies should lead to a new ontology that identifies equivalent elements but that tries to keep unrelated elements apart, as far as this is possible without violating the requirements that are imposed on the structure of an ontology.

As an example, let us consider the following two partial orders:



We assume that some elements of these structures are known to be equivalent. This is expressed by a relation $R \subseteq P \times Q$

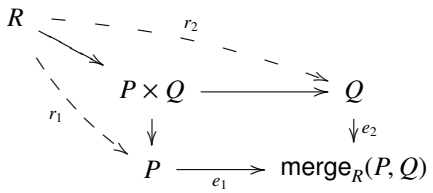


Figure 1: The pushout construction

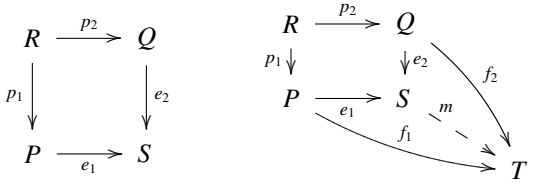
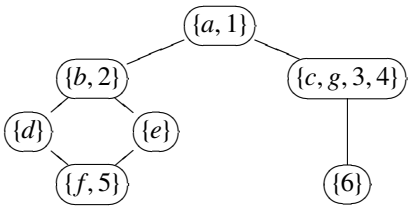


Figure 2: Diagrams for the pushout

(usually called an *ontology mapping*) that we define as $R = \{(a,1),(b,2),(c,4),(f,5),(g,3)\}$. A reasonable result of merging the posets P and Q would then be the following structure:



Observe that all elements related by R are indeed identified, but that some additional identifications are necessary to obtain a partially ordered set. Categorically, we can already specify the data that we have considered for such an operation. The given situation is depicted in Figure 1. The dotted arrows r_1 and r_2 are those that are obtained by composing the projections of the product with the morphism from R to $P \times Q$. They project every pair of elements of R to its first and second component, respectively. Now the result of merging P and Q is not just some poset $\mathsf{merge}_R(P,Q)$, but also the two obvious embeddings of P and Q into $\mathsf{merge}_R(P,Q)$. The property that R-related elements are identified can now be expressed in terms of functions: we find that, for any pair $(p,q) \in R$, $e_1(p) = e_2(q)$. Still a better way to express this for arbitrary morphisms is to say that $e_1 \circ r_1 = e_2 \circ r_2$.

This condition alone, however, does not suffice. Usually, there are many objects for which $e_1 \circ r_1 = e_2 \circ r_2$ holds. Which of these is the one which we want to consider as the *merging* of P and Q? Clearly, the merging shall not identify anything unnecessarily. This can be stated by means of another *universal property*, as follows.

Definition 2 For a category $\mathscr{C}$, consider objects R, P, Q, and morphisms $p_1 : R \to P$ and $p_2 : R \to Q$. An object S together with two morphisms $e_1 : P \to S$ and $e_2 : Q \to S$ is a *pushout* if it satisfies the following properties:

(i) $e_1 \circ p_1 = e_2 \circ p_2$, i.e. the diagram in Figure 2 (left) *commutes*.

(ii) For every other object T and morphisms $f_1 : P \to T$ and $f_2 : Q \to T$, with $f_1 \circ p_1 = f_2 \circ p_2$, there is a unique morphism $m : S \to T$ such that $f_1 = e_1 \circ m$ and $f_2 = e_2 \circ m$. This situation is depicted in Figure 2 (right).

Condition (ii) in this definition states the universal property of the pushout, requiring that it is in a sense the most general object that meets all requirements. Let us try to explain this a bit further. We have already understood that in this setting we can encode the ontology mapping (e.g. binary relation) R conveniently, in that the resulting S identifies (at least) all those elements which are related by R. But now we want to *avoid* the identification of other elements as much as possible. Intuitively, this means that a suitable pushout object needs to keep elements from both components as distinct as possible, while still implementing all necessary identifications, and without including irrelevant information. Enforcing the desired identifications was achieved by condition (i) in the above definition. Excessive identifications are prevented by requiring the *existence* of a factorization m: appending m to e_1 and e_2 cannot make prior identifications undone, and hence a pair that was merged in S can never be separated in an alternative solution (T, f_1, f_2) if a suitable m is known to exist. Finally, the possibility of including entirely unrelated information, like adding some elements not present in either P or Q, is ruled out by assuring *uniqueness* of the factorization m: if S would include elements that are neither in the image of e_1 nor in the image of e_2 then a valid factorization can assign these to arbitrary values in T without loosing the factorization property – but this would result in many possible choices in place of m. In other words, having "unnecessary" elements in the S would result in additional degrees of freedom in the choice of m, thus violating the required uniqueness.

How to put our approach into practice

Let us now see how our approach can be used as a guidance for ontology merging. Decisions need to be made step by step, and we propose the following workflow. Later steps, however, may indicate that earlier decisions need to be revised, and thus to retrace to earlier points.

1. *Decide on ontology representation language used.* This first step is probably the most unproblematic, since there are standard ontology languages around, and the specific application case will usually dictate the language. Potential candidates are e.g. F-Logic (Kifer, Lausen, & Wu 1995) and different variants of OWL (OWL 2004).

2. *Determine what suitable morphisms are.* This step consists of describing the conditions which morphisms must satisfy. These conditions will primarily be dictated by the semantic interpretation of the ontology representation language chosen earlier, and by the specific requirements of the application case. Typical conditions could include the following.

- The preservation of class hierarchies, i.e. functions shall be monotonic with respect to the *general class inclusion* orders on classes and/or roles.
- The preservation of types (e.g. classes, roles, annotated objects).
- The taking into account of model-theoretic logical properties, if featured by the underlying ontology representation language, like satisfiability, or the preservation of specific models.
- The taking into account of proof-theoretic properties, i.e. such relating to particular inference methods chosen for reasoning with ontologies.
- The preservation of language classes, e.g. by requiring that the merging of two OWL Lite ontologies shall not result in an OWL ontology which is not in OWL Lite.

3. *Determine what the ontology mapping is for this setting.* Usually, ontology mappings will be given by (binary) relations between elements of ontologies, indicating which elements shall be identified in the merging process. However, as the product of two ontologies may not always be described conveniently as a set of pairs of elements – as in the case of sets or posets –, it needs to be understood at this stage, what the product really is, and thus what ontology mappings are in this setting.
4. *Determine what pushouts are for this setting.* While the characteristics of a pushout are fully determined by the previous steps, it is still necessary to find a particular instance of the pushout (both for the object and the embedding morphisms) in terms of the ontology language. This requires to define a possible result for arbitrary pushout operations and to show that it satisfies the formal requirements of a pushout. Difficulties at this stage arise from the fact that, like products, pushouts are not guaranteed to exist in general. Negative results may yield effective conditions for the existence of pushouts or even suggest a modification of the considered theory.
5. *Algorithmize how to obtain the mapping.* The issue of how to obtain suitable ontology mappings is a separate issue from the one discussed here, and will usually depend heavily on the application domain and on the ontology representation language chosen. Machine learning techniques may be used here together with linguistics-based approaches (see e.g. (Ehrig & Sure 2004)). Fuzzy relations usually obtained by such approaches may however have to be defuzzified at some stage, in order to obtain a precise ontology mapping which will be used for the merging.
6. *Algorithmize how to obtain the pushout.* At this stage, it is theoretically clear what the pushout – and thus the merged ontology – will be. Casting this insight into an algorithm may require a considerable amount of work. The practitioner may also choose at this step to forego an exact implementation of the merging, and settle for an approximate or heuristic approach for reasons of efficiency, while at the same time being guided by the exact merging result as the ontology to be approximated.

Conclusions

We have argued that the problem of merging ontologies based on a given ontology mapping can be formulated conveniently in the language of category theory. This lead to the well-known definition of the categorical pushout construction, which describes ontological merging independently from the concrete implementation that was chosen. Since pushouts do not exist in all categories, this also yields general guidelines for devising systems of interrelated ontologies. Methods and insights from category theory could be used to assist in the development both of rigorous theoretical settings for ontology merging and of conceptually sound algorithms for practical implementations. Conversely, similar considerations can also be useful to validate merging constructions that have been conceived exclusively on practical grounds, since one may ask in which sense (in which category) a given merging process produces results of general validity.

Acknowledgements We are grateful for helpful comments by the referees on the subject and purpose of this paper. We also acknowledge support by the by the European Commission under contracts IST-2003-506826 SEKT and FP6-507482 KnowledgeWeb, and by the German Federal Ministry of Education and Research (BMBF) under the SmartWeb project. The expressed content is the view of the authors but not necessarily the view of any of the projects as a whole.

References

Ehrig, M., and Sure, Y. 2004. Ontology mapping – an integrated approach. In Bussler, C.; Davis, J.; Fensel, D.; and Studer, R., eds., *Proceedings of the First European Semantic Web Symposium*, volume 3053 of *Lecture Notes in Computer Science*, 76–91. Heraklion, Greece: Springer Verlag.

Goguen, J. 2005. Three perspectives on information integration. In Kalfoglou, Y., and et al., eds., *Semantic Interoperability and Integration*, Dagstuhl Seminar Proceedings 04391.

Jannink, J.; Pichai, S.; Verheijen, D.; and Wiederhold, G. 1998. Encapsulation and composition of ontologies. In *Proceedings of the AAAI Workshop on AI & Information Integration*.

Kent, R. E. 2005. Semantic integration in the Information Flow Framework. In Kalfoglou, Y., and et al., eds., *Semantic Interoperability and Integration*, Dagstuhl Seminar Proceedings 04391.

Kifer, M.; Lausen, G.; and Wu, J. 1995. Logical foundations of object-oriented and frame-based languages. *Journal of the ACM* 42.

2004. Web ontology language (OWL). `www.w3.org/2004/OWL/`.

Schorlemmer, M.; Potter, S.; and Robertson, D. 2002. Automated support for composition of transformtional components in knowledge engineering. Technical Report EDI-INF-RR-0137, Division of Informatics, University of Edinburgh.

Reasoning about multi-contextual ontology evolution

Maciej Zurawski

CISA (Centre for Intelligent Systems and their Applications)
School of Informatics, Edinburgh University
Appleton Tower, Crichton Street,
Edinburgh, EH8 9LE, Scotland, UK
m.zurawski@sms.ed.ac.uk

Abstract

In this paper we develop a formalization and algorithms that can manage the evolution of several ontologies from different contexts, using automated reasoning. It is in general difficult to maintain consistency between several ontologies, but we focus on developing computationally efficient ways of achieving this. Our formalization uses both the notions of several local contexts and of a sequence of states. We believe such a system can become a component in for example a distributed knowledge management system or some other knowledge infrastructure that requires semantic autonomy, i.e. lack of centralized semantics, but presence of a type of semantic coherence. In this paper version we summarize our approach.

Background and motivation

We envision that there will be a need for different kinds of systems that can support several ontologies, their individual evolution and maintain a type of coherence between them. For example, we would like to be able to build systems that will function as organizational knowledge infrastructures. The organizations using these will probably be decentralized and consist of separate divisions that have local autonomy in their knowledge-creating processes. Here we particularly mean semantic autonomy (see the partial definition in figure 1). Such an organization should act as a unified whole, because otherwise entities from outside (e.g. customers) interacting with the organization might be disappointed that it contradicts itself. Creating an organizational knowledge infrastructure is one application area (Zurawski, 2004), but there should exist other applications as well that also requires semantic autonomy. In both cases, this is modeled using several ontologies that can evolve, but where a kind of consistency is maintained between them.

Semantic autonomy requires (among others) these properties to hold at the same time:

1. The local contexts have the freedom to propose a change in their local ontology (i.e. the ontology of the local context).
2. The system does "in some way" maintain global ontological consistency (although it may be the case that the system doesn't have a global theory).
3. The ontological language is dynamic and open-ended (i.e. not confined by a pre-defined set) and there is an oracle (knowledge source) that can answer questions about this language.

Figure 1. In this paper we focus on developing algorithms that exhibit these three properties.

We mention here again very briefly a partial definition of semantic autonomy (for a full definition and detailed discussion about all the requirements, see Zurawski 2004). Semantic autonomy requires the properties in figure 1 to hold. Considering these requirements it is natural that our system should have an explicit notion of states. In the mentioned paper we discussed these requirements and why they make sense, and have to be possessed by distributed knowledge management system (DKM). In this paper we will instead focus on how to actually develop algorithms that have satisfied the requirements listed here.
This paper version is only a short summary of our approach. We don't present the full formalization, but focus more on the motivation.

Notions

We will first explain the basic notions we are using in order to design a system that has the above-mentioned properties.

By *multi-contextual* we mean that we have defined a finite set of subsets that all have their unique identifiers,

individual ontologies and represent a certain cognitive perspective, i.e. an individual way of representing a certain domain. Because different contexts describe the same domain (but using different ontologies) it will be natural that it is possible to create mappings between concepts in the different ontologies. We use the compose-and-conquer type of context-sensitivity (Bouquet et al. 2001).
By *ontology* we currently mean a subsumption hierarchy of logical concepts, that belongs to a certain context (we don't have any global ontology).

By *evolution* we mean that every individual ontology can change. We will particularly focus on the case when a new concept is added to an ontology together with a mapping within its home ontology and a mapping to another ontology. Because ontology evolution is so important, we have defined the notion of *states* that describe in which state a certain ontology is in. Every time an ontology changes it moves from one state to the next one, and all states are ordered.

The logical representation of ontology mappings

Because of limited space we don't define the model theory here, but we however summarize its characteristics. Now we will focus on how to formalize the ontology mappings themselves. The logic we are using has two fundamental dimensions that are used simultaneously: *contexts* and *states.* The basic entities that inhabit this logical space are *concepts*. Every concept belongs to a unique context, and there are no concepts outside the contexts. Every concept has to be created in a certain state and persists either forever or until it is deleted in some state in the future. Concepts can be applied to *instances* and then they act as logical predicates applied to constants. However, in the algorithms that we mention we will only focus on concepts and mappings – no instances will explicitly be present. Two concepts belonging to the same ontology can also be used be combined in these formulas: $R_i \wedge Q_i$ and $R_i \vee Q_i$ (and these can be nested). We have to some extent been inspired by intensional logic (see L. T. F. Gamut, 1991), although we will later focus only on the proof theory and rewrite rules.

State operators and their combination

In order to understand the ontology mapping notation we have understand its three main components, and the first component is the collection of state operators. We don't provide the formal definition here. However, informally speaking, G_r means that something will be true in all future states after r, that F_r means that something will be true at least once in the future after state r and N_r that something is true in state r. We also use a state c from which a state operator is evaluated (i.e. observed).
Moreover, we have defined a way of combining state operators, so that two state operators can be combined into one. The motivation is that this is needed when we want to combine two ontology mappings into one (and all mappings contain state operators as we will see). The reason for introducing the variables r and c is that this becomes practical later for talking about when an ontology mapping was created and in which states it is valid.

Quantifiers and their combination

The second component of ontology mappings is quantifier symbols, and there just two of them: α_1 and α_2. We don't provide the formal definitions here, but α_1 approximately means that we use a universal quantifier and α_2 an existential one. The reason why we have introduced these symbols is that they will be used in the ontology mappings, and can discern the difference between saying that a certain concept is true for all instances or for at least one instance.

Boolean functions and their combination

This is the third component of ontology mappings. We use the standard Boolean functions of two variables and they are represented in 2-DNF form. Two such Boolean functions can be combined by conjunction into a new Boolean function, using standard logical operations.

The ontology mapping notation

Here we show a part of the notation that is used for describing ontology mappings. The reason why we choose this kind of formalization is it that it seems to be good when doing efficient and automated proofs about ontology mappings. Let us call every mapping between two concepts m_i, where i is its unique identifier. A mapping m_i that holds between the concept C_1 in ontology j and the concept C_2 in the ontology k can always be expressed using on of the two following forms:

$$m_i(C_{1j}, C_{2k}) = op(\alpha(f(C_{1j}, C_{2k}))) \text{ or}$$
$$m_i(C_{1j}, C_{2k}) = op(\alpha(f(C_{1j}, C_{2k}))) \wedge op'(\alpha'(f'(C_{1j}, C_{2k})))$$

where
$op \in \{N_a, F_b, G_c\}$, $a,b,c \in S$ (the set of states)
$\alpha(\lambda) \in \{\alpha_1, \alpha_2\}$ and
$f(C_{1j}, C_{2k}) \in \{\langle e_1 \wedge e_2 \rangle, \langle e_1 \wedge \neg e_2 \rangle, \langle \neg e_1 \wedge e_2 \rangle, \langle \neg e_1 \wedge \neg e_2 \rangle\}$
where $e_1 = C_{1j}(x_j)$ and $e_2 = C_{2k}(y_k)$
(the notation of the Boolean function is the set of the conjunctions that a 2-DNF form would contain)

Combining ontology mappings by using the three kinds of rewrite or combination rules

Using the three kinds of rewrite or combination rules, we can now use them in a sequence and use them for

combining any two ontology mappings into one – that is their purpose. The first transformation is the application of rewrite rules for state operators in a way that combines two state operators into one. The second transformation is the application of rewrite rules for expressions with quantifiers in a way two combines to operators into one. The third transformation is the application of combination of Boolean functions in a way that combines two such functions into one.

Examples of mappings

The language mentioned above allows creating a huge variety of mappings. We can for example imitate the five proposed mapping types by Giunchglia (see for example Bouquet 2003) and restate them in this new concise language. Both formalisms use the notion of contexts, but the difference is that our definitions utilize the notion of states as well. The state when a mapping was created is denoted by *r*. Here are some examples:

CORRESPONDENCE – COR(C_{1j}, C_{2k})

$$N_r\alpha_1\{\langle e_1 \wedge e_2\rangle,\langle \neg e_1 \wedge \neg e_2\rangle\} \wedge G_r\alpha_1\{\langle e_1 \wedge e_2\rangle,\langle \neg e_1 \wedge \neg e_2\rangle\}$$

IS (C_{1j}, C_{2k})

$$N_r\alpha_1\{\langle e_1 \wedge e_2\rangle,\langle \neg e_1 \wedge \neg e_2\rangle,\langle \neg e_1 \wedge e_2\rangle\} \wedge$$
$$G_r\alpha_1\{\langle e_1 \wedge e_2\rangle,\langle \neg e_1 \wedge \neg e_2\rangle,\langle \neg e_1 \wedge e_2\rangle\}$$

DISJOINT (C_{1j}, C_{2k})

$$N_r\alpha_1\{\langle \neg e_1 \wedge e_2\rangle,\langle \neg e_1 \wedge \neg e_2\rangle,\langle e_1 \wedge \neg e_2\rangle\} \wedge$$
$$G_r\alpha_1\{\langle \neg e_1 \wedge e_2\rangle,\langle \neg e_1 \wedge \neg e_2\rangle,\langle e_1 \wedge \neg e_2\rangle\}$$

COMPATIBLE (C_{1j}, C_{2k})

$$F_r\alpha_2\{\langle e_1 \wedge e_2\rangle\}$$

(We should stress that for example $\{\langle e_1 \wedge e_2\rangle,\langle \neg e_1 \wedge \neg e_2\rangle,\langle \neg e_1 \wedge e_2\rangle\}$ is equivalent to to $e_1 \rightarrow e_2$, i.e. it is a DNF-form)

For example, the relationship Compatible means "There is at least one future state after r where there is at least a pair of instances (one from ontology j and one from ontology k) where the concept C_{1j} is true (when applied to its instance) at the same time as the concept C_{2k} is true (when applied to its instance)".

Algorithms for verifying consistency between ontology mappings

The problem we are trying to solve can be described as the following proof task. Given a set of existing ontology mappings $m_1(C_{ax}, C_{by})$, $m_2(C_{ax}, C_{by}) \ldots m_n(C_{ax}, C_{by})$, how can we prove if it is consistent the additional mapping $m_{n+1}(C_{ax}, C_{by})$ or not? The variables *x* and *y* refer to the ontologies of the concepts and *a* and *b* are unique concept identities (note that all these variables can be different for every mapping).

To be able to address this proof task we need to have operators that let us express the following things: $m_x \wedge m_y$, $m_x \Rightarrow m_y$ and $\neg m_x$ (and formulas that this can generate). Please note that this language is different from the one that was defined in the beginning (for talking about concepts). Now the basic entity is a mapping.

Then we need two algorithms (called A and B) that will build proof trees using refutation proofs and breath-first search (for proofs), for solving the proof task mentioned. Because of limited space we don't write them down here in full detail, but the algorithms returns an answer (yes/no) each to the following questions:

Algorithm A - "Is mapping G inconsistent with the current mappings?" *Output*: yes/no

Algorithm B - "Is mapping G valid, because it can be inferred from existing mappings?" *Output*: yes/no

This means that in both cases G is the newly proposed mapping, and there is a set of existing mappings (i.e. these are the inputs to algorithms). The algorithms are used in the following way. A newly proposed ontology mapping G is given and first we run algorithm A. If it answers "yes", then we know it is inconsistent with the existing ones. If it answers "no" we run algorithm B. If that algorithm answers "yes" then we know that the newly proposed is valid because it can be inferred from existing mappings (i.e. redundant in some way), and it answers "no" then the proposed mappings is consistent with the existing ones, but can't be inferred from them. Therefore, by using these two algorithms we have covered all three possible cases.

We don't provide here a proof of correctness and completeness. However, we just want to mention that our proof search procedure for refutation proofs using a breadth-first search, and the language used are horn clauses (since we use conjunction, implication and negation). So if the procedure finds a proof, it is valid, and if there is a proof, the procedure will find the shortest one.

Applying the algorithms to ontology evolution

Once both algorithms are in place, it is actually rather straightforward to use them for ontology evolution. An ontology transformation has to be translated to "one or more ontology mappings that are proposed to be added". For example, the addition of a new concept can be seen as inventing a new concept in an ontology and adding an internal mapping (within its home ontology) and a mapping to an external ontology. Then we run algorithms

A and B for both these proposed mappings, and only if there is no created inconsistency detected in neither of the cases, the evolutionary step is accepted and the ontology changes to a new state. Otherwise, the evolutionary step would be forbidden, and the ontology would remain unchanged.

Related research

Background to multi-context logic is give by (Giunchiglia 1993) and multiple languages and bridge rules are discussed. A description and motivation of cognitive context is given by (Giunchiglia et al. 1997) and the notions of locality and compatibility are discussed. In the interesting paper by (McGuiness et al. 2004) automated reasoning using SAT-solvers for class hierarchies is discussed. That is a separate case from the one we are investigating, because WordNet is not an ontology in the sense that there is a strict subsumption relationship between all connected terms. The paper by (Serafini et al., 2003) also investigates semantic matching using SAT and class hierarchies. Some of the inspiration how to design and formalize our logical representation comes from (Gamut, 1991) that describes intensional logic. A variety of different ontology-change operations are classified and described by (Noy & Klein, 2004). Much of the motivation why we need a system that can evolve multiple ontologies is given in (Zurawski 2004).

Conclusions

It will be important for many applications to be able to support many ontologies, that all can evolve at the same time as consistency is maintained between them. We have proposed an approach that uses a logical formalization that consists both of contexts and of states. Every local context has its own individual ontology, and it can evolve – this moves it into the next state. We have already implemented a part of the system (in Java) and when the whole system will be implemented we will evaluate the scalability by running some experiments. Our approach is an alternative to the model theoretical approach where SAT-solvers are used. Many of the systems described in the literature usually only allow for a few types of ontology mappings whereas our ontology mapping language is relatively rich. We have to investigate how this approach compares to other approaches (such as SAT-solving) and investigate how well it scales in cases when there are extensive amounts of ontology mappings that have to be taken into account. The problem of maintaining consistency between multiple evolving ontologies might seem to be intractable, but by adapting the reasoner to the unique properties of the problem, we might make the problem tractable (but experimental evaluation is needed as well). Finally, we believe that these methods could become one of the components in the design of an organizational distributed knowledge management system or some other knowledge infrastructure that will become valuable in the upcoming era of the knowledge society.

Acknowledgements

This research was funded by the Marcus Wallenberg Foundation for Education in International Industrial Enterprise. The author would like to thank Dave Robertson and Jessica Chen-Burger (both at CISA) for their valuable comments and feedback.

References

Bouquet, P., Ghidini, C., Giunchiglia, F., Blanzieri, E., "Theories and uses of context in knowledge representation and reasoning", IRST Technical Report 0110-28, Istituto Trentino di Cultura, October, 2001.

Bouquet, P., Giunchiglia, F., van Harmelen, F.,Serafini, L., Stuckenschmidt, H. "C-OWL: Contextualizing Ontologies", *Proceedings of the Second International Semantic Web Conference*, K. Sekara and J. Mylopoulis (Ed.), pp 164-179, LNCS. Springer Verlag, October, 2003.

Gamut, L. T. F. "Logic, Language, and Meaning, Volume 2: Intensional Logic and Logical Grammar". The University of Chicago Press, 1991.

Giunchiglia, F., "Contextual reasoning", In: *Epistemologia - Special Issue on I Linguaggi e le Macchine*, XVI, pp 345-364, 1993.

Giunchiglia F., Bouquet P., "Introduction to contextual reasoning. An Artificial Intelligence Perspective", In: *Perspectives on Cognitive Science*, B. Kokinov (ed.), 3, NBU Press, Sofia (Bulgaria), 1997.

McGuiness, D. L., Shvaiko, P., Giunchiglia, F., da Silva, P. P., "Towards explaining semantic matching". Technical Report DIT-04-019, Informatica e Telecomunicazioni, University of Trento, 2004.

Noy, N. F. and Klein, M., "Ontology evolution: Not the same as schema evolution". In: *Knowledge and Information Systems*, 6(4), pp 428-440, 2004.

Serafini, L., Bouquet, P., Magnini, B., Zanobini, S., "An algorithm for matching contextualized schemas via SAT". Technical Report DIT-03-003, Informatica e Telecomunicazioni, University of Trento, 2003.

Zurawski, M., "Towards a context-sensitive distributed knowledge management system for the knowledge organization", *Workshop on Knowledge Management and the Semantic Web*, 14th International Conference on Knowledge Engineering and Knowledge Management. EKAW 2004, Northamptonshire, UK, October, 2004.

Default Reasoning with Contexts

Daniel B. Hunter, Daniel F. Bostwick

BAE Systems, Advanced Information Technologies
6 New England Executive Park
Burlington, Massachusetts 01803
daniel.hunter@baesystems.com
daniel.bostwick@baesystems.com

Abstract

We describe a system that combines default reasoning with contexts. Contexts are arranged in a hierarchy where more specific contexts represent revisions of the state of belief in more general contexts. We describe our algorithm for default reasoning in a context hierarchy and provide a translation of our representation into a default logic theory whose inferences agree with our algorithm. We conclude with a discussion of different notions of context and give a justification of our default rules when contexts are understood as states of belief.

Introduction

People apply default reasoning all the time; we fill in missing information about a particular situation based on our experience and knowledge of what is usually true. Applying this type of reasoning is useful as it saves us from having to re-obtain information that remains largely static across most similar situations. Default reasoning in formal systems is likewise useful; it saves us from re-representing information that does not (usually) change.

People also believe different things at different times and in different situations; we operate in different states of belief according to changes in the information we have. The ability to revise our states of belief is an essential part of living in a dynamic world. Representing different states of belief in an automatic inference system is also useful; it allows us to do 'what-if' reasoning and can have a positive impact on inference efficiency. In this paper we view contexts as states of belief and investigate how default reasoning and contexts interact in reasoning.

We describe an algorithm for default reasoning in contexts that we have implemented in an automated inference system. We then provide a translation of our representation into a formal theory of default reasoning and show that our algorithm agrees with the theory. Finally we justify the theory when contexts are understood as states of belief.

Default Reasoning with Contexts in AKS

We have designed and implemented the AIT Knowledge Server (AKS), a computationally-efficient, constraint-based knowledge server that provides a semantics richer than conventional frame system attribute/value relations [Minsky, 1975]. In addition, the AKS supports contexts, which partition the knowledge base. Contexts enable reasoning within a subset of the knowledge base, and they allow different parts of the knowledge base to be inconsistent with each other, facilitating "what if" reasoning. In the AKS, contexts are arranged in a single (tree-structured) inheritance hierarchy such that anything that is true in a context is also true in its subcontexts.

The expressions in our representation language that are relevant to default reasoning are:

- *instance(C, I, K)*: class K is the most specific superclass of instance I in context C
- *parent_class(C, H, K)*: class K is a most specific superclass of class H in context C
- *subcontext(C2, C1)*: context C1 is the most specific supercontext of context C2
- *direct_assignment(C, I, S, V)*: V is assigned to be a value of slot S on instance I in context C
- *default_value(C, K, S, D)*: D is the default value of slot S on class K in context C

Default Reasoning in AKS

The AKS also supports a form of default reasoning in which a slot on a class may be assigned a default value. Default values are inherited by subclasses and instances of the class;

these default values can be overridden by subclasses and instances. In the event that an instance does not specify a value for a slot, then the inherited default value for the slot, if specified, becomes the value of the instance's slot. Slots are not required to have any value assigned to them.

The value for a slot on an instance can either be assigned directly or inherited from a default value in a superclass. If slot S on instance I (notated as I.S) has no directly assigned value, we search upwards in the class hierarchy for a most specific superclass of I that has a default value for S. With contexts, the reasoning becomes more complicated. Consider Figure 1 where we have two contexts:

1. a context called *birds* in which we are agnostic as to whether or not birds can fly, but we know that penguins, in general, cannot
2. a subcontext called *birds_can_fly* in which we have modified our belief to be that birds, in general, can fly

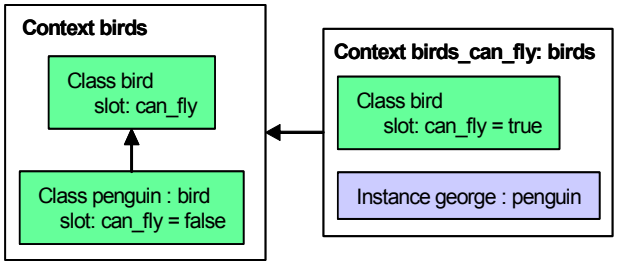


Figure 1. Class hierarchy over two contexts

The question is whether or not *george* can fly in the context *birds_can_fly*. Given a context C, an instance I of class K (where K is a most specific superclass of I), and a slot S on I, we use the following algorithm to determine the value of I.S in C. We use a special symbol *no_value_found* to indicate that no value has been assigned to I.S in C.

The function *value_of(C, I, S)* returns a value for slot S on instance I in context C. *value_of* first looks for a value that is directly assigned to I.S in C or any supercontexts of C. Failing that, *value_of* looks for a default value on slot S in the superclass of I in context C.

```
value_of(C, I, S ):
    V = direct(C, I, S)

    if V == no_value_found
        return default(C, K, S)
    else
        return V
```

direct(C, I, S) returns a value that is directly assigned to slot S on instance I in context C or any ancestor contexts of C.

```
direct( C, I, S ) :
    if direct_assignment(C, I, S,V ) then
        return V
    else
        if subcontext( C, Cp )
            return direct(Cp, I, S )
        else
            return no_value_found
```

The function *default(C, K, S)* returns a default value for slot S on class K in context C. The context hierarchy is searched before the class hierarchy. All ancestor contexts of C are searched for a default value on K.S before any superclasses of K are searched.

```
default(C, K, S ) :
    V = default_for_class(C, K, S )
    if V != no_value_found
        return V
    else
        for each Kp where parent_class(C, K, Kp )
            V = default(C, Kp, S )
            If V != no_value_found
                return V
        return no_value_found
```

The function *default_for_class(C, K, S)* returns a default value for slot S on class K in context C. *default_for_class* does not examine any superclasses of K; it searches only in context C and its supercontexts for a default value on K.S.

```
default_for_class(C, K, S ) :
    if default_value(C, K, S, V )
        return V
    else
        if subcontext( C, Cp )
            return default_for_class(Cp, K, S)
        else
            return no_value_found
```

For the situation illustrated in Figure 1 this algorithm gives the result:

value_of(birds_can_fly, george, can_fly) = false

Justification of Default Reasoning Mechanism

[Etherington, 1988] provides a translation of a class inheritance hierarchy with exceptions into a set of default rules in Reiter's system of default logic. A similar translation can be given for our system of default reasoning.

In Reiter's system of default logic [Reiter, 1980], a *default theory* is a pair $\Delta = \langle D, W \rangle$, where W is a set of first-order formulas and D is a set of rules of the form:

$$\frac{\alpha(\bar{x}) : \beta(\bar{x})}{\chi(\bar{x})}$$

where $\alpha(\bar{x}), \chi(\bar{x})$, and $\beta(\bar{x})$ are first-order formulas whose free variables are among $\bar{x}$.

W is the set of facts and D provides a means of drawing tentative conclusions. The intuitive meaning of the above default rule is that if $\alpha(\bar{a})$ is known and it is consistent to believe $\beta(\bar{a})$, then $\chi(\bar{a})$ may be inferred. ($\bar{a}$ is a sequence of individual constants replacing the variables $\bar{x}$.)

A more formal interpretation of a default theory is provided by the notion of an *extension*. An extension of a default theory <D, W> is a set E of formulas that is a minimal fixed point[1] of an operator Γ on sets of formulas satisfying:

1. $W \subseteq \Gamma(S)$
2. $\Gamma(S)$ is closed under logical consequence
3. Given $\alpha(\bar{x}):\beta(\bar{x})/\chi(\bar{x}) \in D$, if $\alpha(\bar{a}) \in \Gamma(S)$ and $\neg\beta(\bar{a}) \notin S$, then $\chi(\bar{a}) \in \Gamma(S)$

An extension for a default theory is often regarded as a set of propositions constituting an "acceptable" set of beliefs given the theory.

A default rule with $\beta(\bar{x}) = \chi(\bar{x})$ is said to be *normal*; one with $\beta(\bar{x}) = \chi(\bar{x}) \wedge \varphi(\bar{x})$ for some $\varphi(\bar{x})$ is *semi-normal*. Our translation will always result in either normal or semi-normal default rules.

If Σ is a set of AKS assertions describing an inheritance hierarchy with contexts, we give a translation of Σ into a default theory <D(Σ), W(Σ)> as follows:

1. If instance(C, I, A) ∈ Σ, then $C \rightarrow A(I) \in W(\Sigma)$
2. If parent_class(C, B, A) ∈ Σ, then $(x)(C \wedge B(x) \rightarrow A(x)) \in W(\Sigma)$
3. If subcontext(C2, C1) ∈ Σ, then $C2 \rightarrow C1 \in W(\Sigma)$
4. If direct_assignment(C,I,S,V) ∈ Σ, then the default $C:S(I,V) \wedge \neg C_1 \wedge \ldots \neg C_k / S(I,V) \in D(\Sigma)$, where the C_i are *nearest* subcontexts of C such that direct_assignment(C_i, I, S, V′) ∈ Σ for some V′ (i.e. there is no context between C and C_i in which a direct assignment to S for I is made).
5. If default_value(C, K, S, D) ∈ Σ, then $C \wedge K(x):S(x,D) \wedge \neg E_1 \wedge \ldots \neg E_k / S(x,D) \in D(\Sigma)$, where the E_i are all the *exceptions* (defined below) to default_value(C, K, S, D).

The exceptions to default_value(C, K, S, D) are the following:

i. If direct_assignment(C′, I, S, V) ∈ Σ, where C′ is C or a subcontext of C and I is an instance of K, then $(C' \wedge x = I)$ is an exception.

ii. If direct_assignment(C′, I, S, V) ∈ Σ, where C′ is a supercontext of C and I is an instance of K, then $x = I$ is an exception.

iii. If default_assignment(C′, H, S, D′) ∈ Σ, where C′ is a subcontext of C and H is K or a subclass of K, then C′∧H(x) is an exception.

iv. If default_assignment(C′, H, S, D′) ∈ Σ, where C′ is a supercontext of C and H is a subclass of K, then H(x) is an exception.

Given this translation, the algorithm for default reasoning described in the previous section can be shown to agree with the conclusions derivable from <D(Σ), W(Σ)>. More precisely, we have the following theorem:

Theorem 1. *If AKS infers conclusion P from default theory <D(Σ), W(Σ)>, then P is in an extension of <D(Σ), W(Σ)>*[2].

What Contexts Represent

Applying the translation to the example in the previous section yields the following default theory:

W = { birds_can_fly → birds, birds_can_fly → penguin(george) }

D = { birds∧penguin(X) : can_fly(X,false)/can_fly(X,false),

birds_can_fly∧bird(X) : can_fly(X,true) ∧¬penguin(X)/

can_fly(X,true) }

It is straightforward to show that can_fly(george,false) is derivable from birds_can_fly.

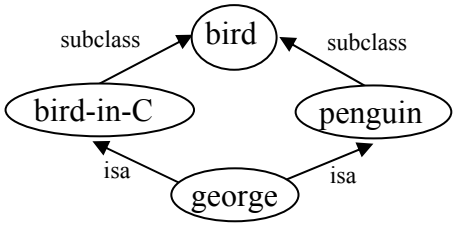


Figure 2. Avian inheritance hierarchy.

As the above default theory shows, we regard penguins as exceptions to birds flying even in subcontexts of the context in which penguins are declared not to fly by default; yet we don't regard birds in the context birds_can_fly as special kinds of birds whose default properties can override those of other subclasses of bird. This introduces an asymmetry in the treatment of subclasses and subcontexts. We might have regarded contexts as defining special subclasses of the classes defined within them, in which case we could

[1] X is a fixed point of an operator O if O(X) = X.

[2] For reasons of space, proofs could not be included in this paper. They are available upon request from the authors.

represent the entire class/context hierarchy in the previous section in terms of a single class hierarchy with no explicit contexts as shown in Figure 2. The context information is embedded in a special subclass of bird, the class bird-in-C (where C stands for a subcontext of the context birds).

If the default flying status for a bird-in-C is that it flies, then, we cannot arrive at a clear conclusion about George's flying ability. (This example is isomorphic to the notorious "Nixon diamond" [Touretzky, 1984].)

We wish to argue that whether or not any inference can be made about George's flying abilities given the above default rules depends upon how contexts are interpreted. [Akman and Surav, 1996] surveys the multifarious ways in which the notion of context has been understood. Some think of contexts as being the same or similar to *situations* as understood, say, in situation theory [Barwise and Perry, 1983]. A situation is regarded as a collection of states of affairs, where states of affairs might be thought of as possible facts. An actual situation might even be identified with a particular spatio-temporal slice of the universe.

If contexts are thought of as situations, then it does make sense to regard the restriction of a class to those instances occurring in a particular context as a subclass of that class[1]. For on this construal of contexts, C2 is a subcontext of C1 if C2 is a restriction of C1 to some particular subset of the facts occurring in C1. Thus the set of instances of a class K that occur in context C2 will be a subset, and often a proper subset, of the set of instances of K occurring in C1. For example, if by default penguins don't fly and one considers, say, birds-inhabiting-Patagonia, which do fly by default, there's no reason to simply assume that penguins-inhabiting-Patagonia don't fly. (Perhaps unusual gravitational conditions in Patagonia give all birds the ability to fly.)

Another interpretation of "context" is as a state of belief. On this interpretation a subcontext represents an *extension* of its parent's state of belief. Thus a subcontext represents a state of belief in which all of the beliefs in the parent are still held plus other beliefs that are added in the subcontext (provided the new beliefs are consistent with the old ones).

On this interpretation of what a context is, defaults holding in supercontexts are inherited by a context unless overridden. Hence the context birds_can_fly simply adds to the information present in its parent context birds, the information that by default birds can fly. This default rule is consistent with the information contained in birds that by default penguins cannot fly and so that piece of information is inherited by birds_can_fly. The default inference rules for AKS class/context hierarchies can therefore be justified when contexts are understood as belief states[2].

Conclusion

We have described an implementation of a default reasoning system that combines class inheritance hierarchies with contexts. A translation of assertions in our system into default rules in Reiter's system of default logic was given. These default rules can be justified when the notion of a context is understood as a state of belief and a subcontext as an extension or revision of a state of belief.

References

Akman, V. and Surav, M. 1996. "Steps Toward Formalizing Context," in *AI Magazine* 17(3), pp. 55-72,

Barwise, J. and Perry, J. 1983. *Situations and Attitudes*. Cambridge:MIT Press.

Etherington, D. W., 1988. *Reasoning with Incomplete Information*. Los Altos: Morgan Kaufmann Publishers.

Minsky, M. 1975. "A framework for representing knowledge" in P. Winston ed., *The Psychology of Computer Vision*. New York: McGraw-Hill, pp. 211-280.

Reiter, R. 1980. "A logic for default reasoning" in *Artificial Intelligence 13*, North-Holland, pp. 81-132.

Shore, J. "Relative Entropy, Probabilistic Inference, and AI." In *Proceedings of the First Conference on Uncertainty in Artificial Intelligence*, ed. L. Kanal and J. Lemmer, Elsevier Science Publishing Co., Inc., 1985.

Touretzky, D. S., 1984. "Implicit ordering of defaults in inheritance systems." In *Proceedings of the Fifth National Conference on Artificial Intelligence*, pp. 322-325.

Acknowledgement

This material is based upon work supported by the Air Force Research Laboratory under Contract No. F30602-01-C-0041. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the United States Air Force.

[1] In Cyc™, for example, it is assumed that quantifiers in rules for a given microtheory range only over objects existing in the situation described by the microtheory.

[2] A more formal argument for this conclusion can be given, using maximum relative entropy updating for belief revision and a probabilistic/statistical model of default rules.

Architecting a Search Engine for the Semantic Web

David E. Goldschmidt and Mukkai Krishnamoorthy

Rensselaer Polytechnic Institute
Troy, New York, USA
{goldsd, moorthy}@cs.rpi.edu

Abstract

Since its emergence in the early 1990s, the World Wide Web has rapidly evolved into a global information space of incomparable size. Keyword-based search engines such as Google™ index as many webpages as possible for the benefit of human users. Sophisticated as such search engines have become, they are still often unable to bridge the gap between HTML and the human. Tim Berners-Lee envisions the *Semantic Web* as the web of machine-interpretable information that complements the existing World Wide Web, providing an automated means for machines to truly traverse the Web on behalf of their human counterparts. A cornerstone application of the emerging Semantic Web is the search engine that is capable of tying components of the Semantic Web together into a traversable landscape. This paper describes both an architecture for and a prototype of a *Semantic Web Search Engine* (*SWSE*) using Jena that provides more sophisticated searching with more exacting results. To compare keyword-based search via Google with semantics-based search via the SWSE prototype, we utilize the *Google CruciVerbalist* (*GCV*), a system we developed that attempts to solve crossword puzzles via a generic search interface.

Introduction

For many, a search engine is the starting point for locating new information on the Web. Among companies specializing in Web search technologies, Google currently enjoys a top spot in terms of both coverage and reliability (Elgin 2004). Google's search technologies rely on the linked structure of the Web to rank webpages based on their popularity. Other search engines use a variety of word-frequency and clustering techniques, as well as additional keyword-based approaches. Regardless of the underlying architecture, users specify keywords that match words in huge search engine databases, producing a ranked list of URLs and snippets of webpages in which the keywords matched.

While such technologies have been successful, users are still often faced with the daunting task of sifting through multiple pages of results, many of which are irrelevant. Surveys indicate that almost 25% of Web searchers are unable to find useful results in the first set of URLs that are returned (Roush 2004). Such results are designated for human consumption rather than machine processing.

Tim Berners-Lee, the inventor of the World Wide Web, defines the *Semantic Web* as "The Web of data with meaning in the sense that a computer program can learn enough about what the data means [in order] to process it" (Berners-Lee 1999). Rather than a Web filled only with human-interpretable information, Berners-Lee's vision includes an extended Web that incorporates *machine-interpretable* information, enabling machines to process the volumes of available information, acting on behalf of their human counterparts (Fensel et al 2003).

In this paper, we present the building blocks of the Semantic Web and describe a scalable architecture for a *Semantic Web Search Engine* (*SWSE*) using Jena. A prototype implementation of the SWSE is also presented, including sample ontologies and results. Keyword-based search results from Google are compared to SWSE search results via the *Google CruciVerbalist* (*GCV*), a system developed to solve crossword puzzles using only the results of Google or another such search engine interface (Goldschmidt and Krishnamoorthy 2004).

Building Blocks of the Semantic Web

Much of the infrastructure of the Semantic Web has already been defined (see Berners-Lee et al 2001, Fensel et al 2003, Hjelm 2001, Heflin et al 2002, and others).

Uniform Resource Identifier (URI). *Uniform Resource Identifiers* (*URIs*) are used to represent tangible objects, people, places, abstract relationships, intangible or fuzzy concepts—just about anything (Connolly 2003). Syntactically, URIs resemble URLs. Defining URIs enables the development of ever-expanding machine-interpretable vocabularies. These are the nouns, verbs, and other language constructs that make up the Semantic Web.

Resource Description Framework (RDF). The *Resource Description Framework* (*RDF*) is used to combine URIs together to form machine-interpretable statements. Akin to simple prose, an RDF statement consists of a subject, a predicate, and an object. In general, the subject is a

resource, the predicate is a property, and the object is either a resource or a literal value (see Lassila and Swick 1999, Manola and Miller 2002).

RDF Schema (RDFS). Using basic RDF constructs, rich machine-interpretable vocabularies may be developed. *RDF Schema* (*RDFS*) is an example of a widely used vocabulary language based on RDF that allows you to define classes, subclasses, properties, and subproperties (see Brickley and Guha 2002).

Web Ontology Language (OWL). Incorporating RDF and RDF Schema, the *Web Ontology Language* (*OWL*) further enriches the family of ontology languages available for use on the Semantic Web. OWL provides such constructs as: (1) relations (e.g. equivalence, disjointness); (2) cardinality; (3) richer typing; (4) characteristics of properties (e.g. symmetry, transitivity); and (5) enumerated classes (see McGuinness and van Harmelen 2004).

Ontologies. Using the aforementioned languages, domains of knowledge called *ontologies* are defined. An ontology "formally defines a common set of terms that are used to describe and represent a domain," thus making the terms and knowledge therein reusable (Fensel et al 2003).

Jena Framework. Designed and implemented by Brian McBride et al of HP Labs, the *Jena Framework* is a set of Java APIs devoted to Semantic Web application development. Jena supports ontologies developed using RDF, OWL, and RDFS. Based on these ontology languages, Jena provides a reasoning subsystem that supports RDFS and the *OWL Lite* subset, though OWL support is described as being "preliminary and still under development" (McBride et al 2005). Since the given ontology languages allow the specification of constraints, a means of validating an RDF model is provided via Jena's validation inference interface.

Sample Ontologies

Organized ontologies and other RDF documents are currently being created to support Semantic Web applications. Not surprisingly, much of these efforts are catalogued on the World Wide Web (see SchemaWeb 2005, Swoogle 2005). We constructed numerous ontologies as part of our SWSE implementation, including vocabularies for genealogy, mythology, United States Presidents, the Solar System, geography, and so on.

WordNet® Ontology. In an effort to bring a voluminous and practical vocabulary to the Semantic Web, we looked to *WordNet*®, an open lexical reference of over 200,000 English words and phrases, including their semantic relationships with one another. WordNet has been developed by the Cognitive Science Laboratory at Princeton University under the direction of Professor George A. Miller (WordNet 2005).

In early 2001, Melnik and Decker converted the WordNet vocabulary to RDF and RDFS (Melnik and Decker 2001). We have further translated WordNet to OWL, enabling the use of the WordNet vocabulary as both resources and properties. WordNet defines nouns, verbs, adverbs, and adjectives. We translated these vocabulary constructs to OWL as both properties and resources, appending the corresponding part of speech to distinguish usage. For example, the concept of life is translated to the `life-Noun` resource and the `life-Noun-as-Verb` property, enabling specific use as subject, predicate, or object in RDF statements.

Semantic Webgraphs. Information on the Semantic Web may be represented via *semantic webgraphs*. Such constructs are graphs in which resources and literal values are represented as nodes, and properties as either nodes (see Figure 1) or directed edges (see Figure 2). Semantic webgraphs provide a graph-based human-readable format, and enable software agents to traverse the Semantic Web via well-known graph traversal algorithms.

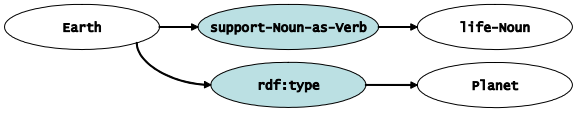


Figure 1. Semantic webgraph showing Earth's support for life

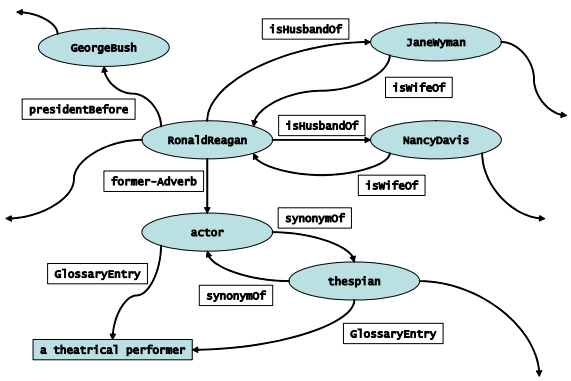


Figure 2. Semantic webgraph focused on Ronald Reagan, including vocabulary from the genealogy, WordNet, and Presidents ontologies

Implementing a Semantic Web Search Engine

SWSE Architecture and Prototype

From a bird's-eye view, the architecture of the *Semantic Web Search Engine* resembles that of traditional keyword-based search engines. Queries are accepted and results generated based on summarized data in a central database.

Search Queries. The basic search query form is plaintext, which supports intuitive features much like that of Google. Text may be quoted to treat multiple words as a single unit, and *stop words* (e.g. "is," "the," "of," etc.) will—to some degree—be ignored. Starting with a plaintext query form maximizes the flexibility of future search enhancements.

Using Search Keywords to Identify URIs. In the SWSE architecture, keyword-based search still plays a role. Given a search query Q in plaintext form, phrases of Q are matched via case-insensitive string matching against the `<rdfs:label>` and `<rdfs:comment>` elements of all available RDF documents, as well as all `<rdfs:Literal>` elements, as identified by property definitions. Results of this string-matching phase are URIs of both properties and non-properties, weighted according to frequency.

Forming RDF Queries. Once potential properties and resources are identified, they are combined to form RDF queries against the RDF knowledge base. This is the heart of the SWSE architecture in which the various combinations of properties and resources are queried and results collected and ranked.

As an example, if a given plaintext query results in potential properties p_1 and p_2, and potential non-property resources n_1, n_2, and n_3, a query will be generated for each combination in which either zero or one element is missing (e.g. $n_1\ p_1\ n_2$; $n_1\ p_1\ n_3$; $n_1\ p_1\ ?$; $?\ p_2\ n_1$; etc.). This "fill-in-the-blank" RDF statement detection process is repeated with those new elements discovered forming a "hop" in the sense of a breadth-first search algorithm.

Given example query "wife of President before Adams," we match substrings against the SWSE knowledge base. From the genealogy ontology, we match the `isWifeOf` property; from the US Presidents ontology, we match the `presidentBefore` and `presidentAfter` properties, as well as the `JohnAdams` and `JohnQuincyAdams` resources. During the first pass, we detect statements shown in Figure 3.

```
GeorgeWashington presidentBefore JohnAdams.
JamesMonroe presidentBefore JohnQuincyAdams.
ThomasJefferson presidentAfter JohnAdams.
AndrewJackson presidentAfter JohnQuincyAdams.
AbigailSmith isWifeOf JohnAdams.
LouisaJohnson isWifeOf JohnQuincyAdams.
```

Figure 3. Detected RDF statements shown in weighted order

For each new resource detected, the process repeats itself, yielding new RDF statements (see Figure 4).

```
MarthaCurtis isWifeOf GeorgeWashington.
ElizabethKortwright isWifeOf JamesMonroe.
MarthaSkelton isWifeOf ThomasJefferson.
RachelRobards isWifeOf AndrewJackson.
JamesMonroe presidentAfter JamesMadison.
ThomasJefferson presidentBefore JamesMadison.
etc.
```

Figure 4. RDF statements detected during the second pass

Though this process may be repeated many times, we limit the number of repetitions—i.e. the maximum breadth—to a small number such as two or three, otherwise results will be flooded with irrelevant inferences.

Traversing Semantic Webgraphs. The RDF statements discovered via the aforementioned querying process form a semantic webgraph. In an attempt to match multiple RDF statements to a given plaintext query, we number each word of the plaintext query, as shown in Figure 5.

```
wife of President before Adams
1    2  3         4      5
```

Figure 5. Sample plaintext query with order specified

We then count the number of potential properties, n_p, detected for the plaintext query string, deduplicating based on location. For the given example, the word "wife" in location `1` matches the `isWifeOf` property, whereas the word "President" in location `3` matches both the `presidentAfter` and `presidentBefore` properties. Counting location `3` only once, n_p is two, indicating that our results should combine at most two RDF statements. More specifically, we traverse only two properties in our semantic webgraph. See Table 1 for example results.

SWSE Prototype Results

As described above, we have successfully implemented and tested an SWSE prototype. Rather than use a relational database, the SWSE prototype stores all of its knowledge in memory.

Sample SWSE Query Results. With the aforementioned ontologies, the SWSE prototype provides results to queries in a fashion reminiscent of Prolog. Example results appear in Table 1.

Plaintext Query	Top SWSE Results
Wife of President before Adams	Martha Curtis | Is Wife Of | George Washington | President Before | John Adams. Elizabeth Kortwright | Is Wife Of | James Monroe | President Before | John Quincy Adams.
Daughter of wife of Norse God of Mischief	Hel | Is Daughter Of | Angrboda | Is Wife Of | Loki | Is God Of | Mischief.
Who wrote the Gettysburg Address?	Abraham Lincoln | wrote | Gettysburg Address. Abraham Lincoln | penned | Gettysburg Address.

Table 1. Example query results using SWSE

Google CruciVerbalist. A fundamental goal of the Semantic Web is to enable machines to communicate with one another via machine-processable vocabularies. In an effort to compare keyword-based search and semantics-based search, we constructed the *Google CruciVerbalist* (*GCV*), a system that attempts to solve crossword puzzles using Google or the SWSE prototype to answer clues.

GCV utilizes numerous keyword-based "tricks" to translate crossword puzzle clues into "Google-friendly" or "search-friendly" query strings. For each query string, candidate answers are obtained from the list of top ten

query results. None of the actual HTML pages are fetched (Goldschmidt and Krishnamoorthy 2004).

Comparative Crossword Puzzle Results. A set of theme-based children's crossword puzzles were used to compare keyword-based searching via Google to semantics-based searching via the SWSE prototype.

As shown in Table 2, the keyword-based approach yields many irrelevant results, whereas the semantics-based approach is much more exacting.

Crossword puzzle	Candidate answers per clue via Google (min / avg / max)	Candidate answers per clue via SWSE (min / avg / max)
Norse Mythology	15 / 68.44 / 124	1 / 1.36 / 3
US Presidents	25 / 107.69 / 206	1 / 5.31 / 12
Solar System	34 / 89.45 / 192	1 / 3.09 / 8
US Geography	4 / 42.00 / 72	1 / 1.10 / 2

Table 2. Comparing the number of candidate answers per clue

Given the set of candidate answers for each clue, GCV attempts to fill in the crossword grid. Each candidate answer is assigned a confidence value based on word frequency; such confidence values drive the depth-first search algorithm used to populate the grid. The fewer incorrect candidate answers, the higher the success rates, as shown in Table 3.

Crossword puzzle	Correctly placed words via Google	Correctly placed words via SWSE
Norse Mythology	9 / 25 (36%)	25 / 25 (100%)
US Presidents	8 / 13 (62%)	12 / 13 (92%)
Solar System	11 / 11 (100%)	11 / 11 (100%)
US Geography	4 / 10 (40%)	10 / 10 (100%)

Table 3. Comparing the success of solving crossword puzzles

Conclusions

Though Google searches occur by the thousands every second (Elgin 2004), technologies for searching the World Wide Web are reaching a plateau. New developments and advancements in keyword-based search technologies will continue to improve search services on the Web; however, the growth rate of these improvements will likely be slight. Problems of imprecise and irrelevant results will continue to hinder Web searchers, especially with the continued expansion of the Web.

A new, semantically based approach is necessary not only to reduce the "information overload" problem of the day, but also to enable more effective and productive services over the Web. By providing a viable architecture and prototype for a Semantic Web search engine, our research aims to help open the floodgates of the emerging Semantic Web.

References

Berners-Lee, T. 1999. *Weaving the Web: the Original Design and Ultimate Destiny of the World Wide Web by Its Inventor*. New York: HarperSanFrancisco.

Berners-Lee, T. et al 2001. The Semantic Web. *Scientific American*. May 2001.

Brickley, D. and Guha, R. eds. 2002. RDF Vocabulary Description Language 1.0: RDF Schema. W3C, http://www.w3.org/TR/rdf-schema.

Calishain, T. and Dornfest R. 2003. *Google Hacks: 101 Industrial-Strength Tips & Tools*. Sebastopol, Calif.: O'Reilly & Associates, Inc.

Connolly, D. et al 2003. Web Naming and Addressing Overview. W3C, http://www.w3.org/ Addressing.

Elgin, B. 2004. Why the world's hottest tech company will struggle to keep its edge. *BusinessWeek*. May 3, 2004.

Fensel, D. et al eds. 2003. *Spinning the Semantic Web: Bringing the World Wide Web to Its Full Potential*. Cambridge, Mass.: MIT Press.

Goldschmidt, D. and Krishnamoorthy, M. 2004. Solving Crossword Puzzles via the Google API. In *Proceedings of the IADIS International Conference WWW/Internet 2004*, 382-389. Madrid, Spain: IADIS Press.

Heflin, J. et al eds. 2002. Requirements for a Web Ontology Language. W3C, http://www.w3.org/TR/webont-req.

Hjelm, J. 2001. *Creating the Semantic Web with RDF*. John Wiley & Sons, Inc.

Lassila, O. and Swick, R. eds. 1999. Resource Description Framework (RDF) Model and Syntax Specification. W3C, http://www.w3.org/TR/REC-rdf-syntax.

Manola, F. and Miller, E. eds. 2002. RDF Primer. W3C, http://www.w3.org/TR/rdf-primer.

McBride, B. et al 2005. HP Labs Semantic Web Research, http://www.hpl.hp.com/semweb/.

McGuinness, D. and van Harmelen, F. eds. 2004. OWL Web Ontology Language Overview. W3C, http://www.w3.org/TR/owl-features.

Melnik, S. and Decker, S. 2001. WordNet via RDF, http://www.semanticweb.org/library/.

Roush, W. 2004. Search beyond Google. *Technology Review*. March 2004. http://www.technologyreview.com/articles/print_version/roush0304.asp.

SchemaWeb 2005. http://www.schemaweb.info/.

Swoogle 2005. http://swoogle.umbc.edu/.

WordNet 2005. http://wordnet.princeton.edu/.

SWARMS: A Tool for Exploring Domain Knowledge on Semantic Web

Liang Bangyong, Tang Jie, Wu Gang, Zhang Peng, Zhang Kuo, Xu Hui, Zhang Po, Yan Xuedong, Li Juanzi

Knowledge Engineering Group, Department of Computer Science, Tsinghua University
{liangby97, j-tang02}@mails.tsinghua.edu.cn

Abstract

This paper introduces SWARMS, a tool for exploring domain knowledge in semantic web. By domain knowledge exploration, we mean searching for or navigating the knowledge in a specific domain. We have found, through an analysis of survey result and an analysis of using log data, that requirements for domain knowledge exploration can be grouped into three categories. The categories include knowledge search, schema based navigation, and search results analysis. Traditional methods usually focus on one of the three types, for example, retrieval of 'relevant knowledge' by exploiting full-text retrieval methods. We propose a tool, called SWARMS, for exploring domain knowledge, in which we provide the ability to conduct domain knowledge exploration by the three categories. Specifically, users can conduct search for special kind of knowledge and they can also interact with the tool by navigating the knowledge base. Furthermore, we conduct analysis for the search or navigation results. The tool is applied to the software management domain. We use ontology as the mean for knowledge representation. The paper describes the architecture, features, and component technologies of the tool.

1. Introduction

Domain knowledge management has made significant progress in recent years, particularly after the emergence of Semantic Web. Many knowledge bases are constructed for managing domain knowledge [AMO03]. However, domain knowledge management does not seem to be so successful. One of the most challenges for domain knowledge management is the exploration of domain knowledge.

Several systems have been developed for domain knowledge exploration [NSD01]. However, most of them look on domain knowledge exploration as a problem of either conventional relevance search or knowledge browsing. In relevance search, when users type a query, the system returns a list of ranked 'targets' with the most relevant 'target' on the top. Here, the target can be document or object in the knowledge base. In knowledge navigation, users select the concept that they want to browse and input some specific constraints from the knowledge schema, and the system returns the 'targets' that belong to the concept and satisfy the constraints. Navigation also enables users to navigate to the objects that are 'similar' to the current browsing object.

In this paper, we try to address the domain knowledge exploration in a novel approach. We categorize the requirements for domain knowledge exploration into three categories, i.e. knowledge search, schema based navigation, and search results analysis.

Our proposal first is to take a strategy of divide-and-conquer, and then is to combine them into a unified system. Users can start their exploration on the knowledge base by typing a keywords-based query. The system returns the relevant objects. And then users select what they want to browse. The object is shown in a navigation view, in which users can browse its schema information, its value, and those objects related to it. In this view, users can navigate to other objects related by the help of a graphic user interface. Users can also specify some constraint and search directly in the navigation view. Finally, for the search results, we provide two kinds of analysis on it by using text mining technologies. The former is similarity analysis and the later is knowledge summary. In the paper, we refer to the approach as 'unified domain knowledge exploration'. The advantage of unified domain knowledge exploration lies in that it can accommodate the knowledge search, navigation, and knowledge analysis well. Furthermore, analysis helps users understand the knowledge easier. It is reasonable particularly in domain knowledge management, because in a domain knowledge is usually represented by a knowledge language (e.g. Web Ontology Language OWL) which makes it difficult for users to understand. Knowledge summary aims to represent knowledge by understandable natural language to users. Similarity analysis helps users to locate the similar objects to what they have obtained or to compare the objects in the knowledge base. We have developed a system based on the approach, which is called SWARMS.

The rest of the paper is organized as follows. In section 2, we introduce related works. In section 3, we explain our approach to the problem. In section 4, we describe the main viewpoints of SWARMS to end users and we introduce the architecture and implementation of SWARMS in section 5. Finally the conclusions are made in section 6.

2. Related Works

Knowledge search can be seen as one part of knowledge management. Knowledge search is concerned with finding the 'relevant' knowledge from knowledge base. For

example, Swoogle uses the techniques from information retrieval to build a search center of semantic web resources [DFJ04]. The search results by Swoogle can be ontology file, concepts, properties and instances. The results are not easily understandable for average user. Semantic Search project extends the keyword based search [GMM03]. It can find the instances that do no contain the keywords in the query. The project aims to enhance the traditional search by the semantic search techniques.

Knowledge navigation aims at 'focus+context' navigation in knowledge exploration. The focus means the object that satisfies current criteria (usually specified by user) and the context means the related objects to the current target. For example, Janecek and Pu propose an interactive visualization technique for exploring an annotated image collection [JP03]. The focus and context are considered and the search results provide both of them. Flink(http://prauw.cs.vu.nl:8080/flink/) gives a graphical view of researcher social network. For a researcher, the view displays his interest fields and researchers that have the same interests with him.

3. Our Approach to Domain Knowledge Exploration

The underlying data models in SWARMS are ontology. Domain knowledge base stores the information organized according to the domain knowledge schema predefined by domain experts. Different from traditional search in which the 'target' is only document and the corresponding search task is the retrieval of relevant documents, domain knowledge can have complicated schema. For example, in the software domain we have defined, there are 19 concepts, 109 properties and 2925 instances in total.

The knowledge schema can help users to organize their data well. It presents explicit semantics for the data, which makes it possible for more advanced applications such as reasoning. On the other hand, it has higher requirement for the knowledge exploration. Question Answering is an ideal form for knowledge access. When users type a natural language question or a query (a combination of keywords) as a description of his search criteria, it is ideal to have the machine 'understand' the input and return only the necessary information based on the request. However, there are still lots of research work to do before putting QA into practical uses. In short term, we need consider adopting a different approach.

We have found that we can group the users' needs into three categories. Specifically, when users don't know the knowledge schema or other domain knowledge, they can launch a search process by only typing several keywords. And the system returns all concepts/properties/instances that contains the keywords. Secondly, when users have specific object that they want to search, they can specify the concepts/properties/instances in the knowledge navigation view. They can specify more constraints before conduct the search. Finally, since data in knowledge base is represented by triples, general users without enough domain knowledge may have difficulty to understand it. We propose analysis technique to deal with the problem. We make use of two methods for analysis, i.e. knowledge summary and similarity analysis.

4. SWARMS

Features

Currently, SWARMS provides three types of exploration. 1) Knowledge Search. It searches the concepts, properties and instances in the knowledge base by making use of full-text search technology. 2) Knowledge Navigation. It provides three kinds of navigation, i.e. concept navigation, instance navigation and eagle eye navigation. 3) Search Result Analysis. It summarizes the knowledge into natural language. A text in natural language describing the meaning or the content of the concepts or instances that users select is returned. Users can also use it to find the similar concepts/instances to what they are interested in.

Ontology Definition and Knowledge Base Construction

We define a software ontology[1] by referencing the schema on SourceForge (http://ww.sourceforge.net), one of the biggest open source software development websites.

We have developed a rule-based wrapper to get the data from SourceForge and store them into the knowledge base according to the ontology.

Search View

There are four types of searches in Search View: full-text search (also called 'document' search), Instances Search, Classes Search, and Properties Search. In document search, users type the keywords, and the system returns a list of ranked entities. The entity can be concept, instance, or property. Each entity is assigned a score representing its relevance to the input keywords. We assign the scores using information retrieval model. The returned entities are grouped into concepts, instances, and properties respectively.

As model, we employ VSM (Vector Space Model) [SWY75], which computes the Cosine Similarity between the input keywords and entities in knowledge base. For computing the Cosine Similarity, we need to construct a document for each entity. We extract bag of words for a concept from its name and properties that are related to it and view the bag of words as the document for the concept. For properties, we further divide it into *object* properties and *datatype* properties. Document for both of the properties are defined by words in its name only. For instances, we only consider concept instances. We do not take into consideration of property instances. There are

[1] The ontology is available at http://www.schemaweb.info/schema/SchemaDetails.aspx?id=235

two reasons: almost all property instances are related to one or more concept instances and a preliminary survey indicates that usually user prefers concept instances to property instances. Score of each entity ranges from 0 to 1, where 0 indicates non-relevance and 1 indicates exact match.

In search, given a query, all entities matched against the query keywords are retrieved and presented in descending order of the relevant scores.

Figure 1 shows an example of instance search. There are two tab views: Text Search view and Visual Search view. Here as the search view, we mean the Text Search view, which is the default view in SWARMS. There are four radio buttons corresponding to the four types of searches. The check box "Summary" indicates knowledge summary (we will describe it in detail below). The left window displays the retrieved instances and the right window displays the detailed information for the selected instance. Detailed information of instance includes its name, value (e.g. string or numeric) of related *datatype* property, and value (i.e. another concept instance) of related *object* property. The bottom window is retained for knowledge summary.

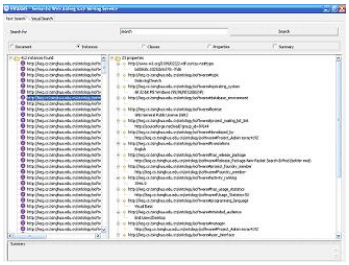


Figure 1. An example of instance search

Navigation View

There are two means to enter the Navigation View: users can double click the entity name in the Search View and users can directly switch to Navigation View by clicking the Navigation View tab.

When users directly switch to Navigation View, the system displays a graph with the concept "Project" in the middle of the view (we think the concept "Project" is a more important concept in software management) and concepts that related to it (as shown in figure 2). In the graph, round node denotes concept, directed edge denotes object property. Users may have different preferences to the concept for navigation. We provide a drill mode for facilitating the navigation. When users are interested in one of the concept, they can double click the round node denoting the concept. A new graph will be rendered which displays the clicked concept in the middle of the graph and surrounds it with concepts that related to it. We have tried displaying all the concepts and relations in the graph, but it results into a very complicated graph that is full of nodes and edges.

Figure 2 shows an example in concept navigation. The main window displays the concept graph, and the top-right window displays properties of the selected concept. The bottom right window is the eagle-eye window. Users can go to any part of the navigation view by selecting the zone in the eagle-eye window.

Figure 3 shows a concept navigation scenario. Users double-click the concept "Project_Admin" or "LatestNew", and then the system returns the corresponding graph that places them in the middle.

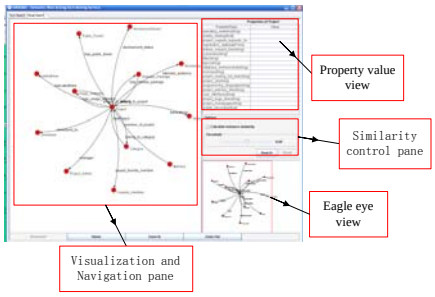


Figure 2. A concept navigation example

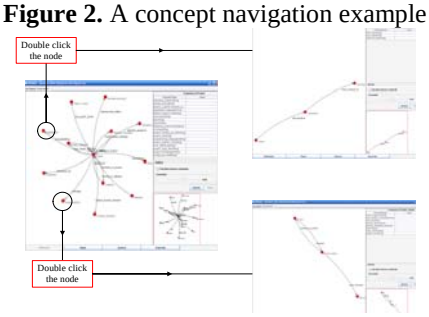


Figure 3. A concept navigation scenario

We also tried to combine knowledge search and navigation into a unified mode. We called it navigation based search. In navigation based search, when users click a concept in the concept navigation view, the top right window list its properties with none values. Then users can input some property values and conduct search by these constraints directly in the navigation view. For example, users may be interested in the projects which are developed by Java language. He can input "Java" in the datatype property "Programming_language", and clicks the "search" button to perform the search. Figure 4 shows the example.

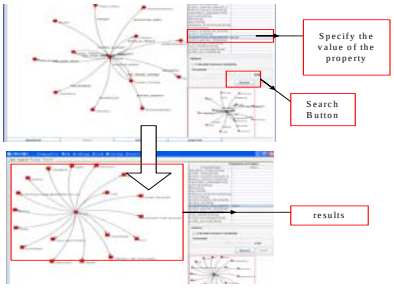


Figure 4. An example of navigation based search

Search Result Analysis

Similarity Analysis

We exploit VSM for computing the similarity between two instances. For instance, we construct the document as we did in the sub-section "Search View". We extract the bag

of words from the 'document' and compute the similarity between two documents by Cosine Similarity method. In similarity analysis, we compute similarity score for every pair of instances of a concept and display the similarity in the graph as shown in figure 5. A similarity threshold slider is placed in the middle of the right window. With the threshold slider, users can control the number of similarity links that displayed in the graph.

Knowledge Summary

Here we conduct the knowledge summary in the interface as an optional function. When search results are displayed, users can select a result and check the "Summary" checkbox. The summary of the entity will be displayed in the summary pane. Figure 5 shows an example summary. The bottom window displays the summary result

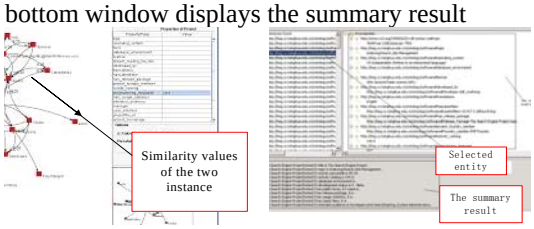


Figure 5. Similarity Analysis and Knowledge Summary

5. Architecture and Implementation

There are six main components in SWARMS: Knowledge Extractor, Domain Knowledge Base, Indexing, Knowledge Search, Navigation, and Search Results Analysis modules.

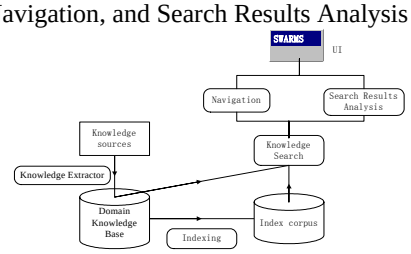


Figure 6. Architecture

We chose SourceForge (http://ww.sourceforge.net) as the knowledge data source. Totally, 1180 software projects are crawled into the knowledge base.

In Indexing module, we derive the ideas from the community of Information Retrieval and build an inverted table indexing. In the inverted table, besides indexing the entities, we also index properties that related to the entities. Indexing for concepts and instances are built independently. For knowledge exploration, we have implemented two kinds of search mechanisms. The first search mechanism makes use of the inverted table indexing. It is aimed for full-text search. The other mechanism is implemented by RDQL(RDF Data Query Language)[Sea03]. It is designed for complicated query. It is appropriate to allow for both high efficiency and advanced search functions.

The Knowledge Search makes use of inverted table indexing. The principle of obtaining the search list and ranking it are described in prior sections. In Navigation, we use both inverted table indexing and RDQL. For navigation based search, we use only RDQL, since the query can be very complicated. The graph visualization in navigation is implemented by JUNG(http://jung.sourceforge.net).

Both similarity analysis and knowledge summary have great computational costs. So they are processed in advance. When new instances come to the knowledge base, the analysis module is called to incrementally calculate the similarity scores among instances and conduct the summary for the new instances. We only calculate the similarity score between any two instances that belong to the same concept

Finally, we provide two kinds of versions: standalone application and web version. They are both available at http://keg.cs.tsinghua.edu.cn/project/pswmp.htm.

6. Conclusion

In this paper, we have investigated the problem of domain knowledge exploration. We have made clear the following issues in the work. 1) Through an analysis, we have found that exploration needs on domain knowledge can be categorized into three types. 2) Based on the finding, we propose a new approach to domain knowledge exploration in which we combine the search, navigation, and search result analysis into a unified method. 3) We have developed a system called 'SWARMS', based on the idea. In SWARMS, we provide features for knowledge search, knowledge navigation, and search result analysis.

References

[AMO03]Angele, J., Mönch, E., Oppermann, H., Staab, S., and Wenke, D. Ontology-Based Query and Answering in Chemistry: OntoNova @ Project Halo. International Semantic Web Conference 2003: 913-928

[DFJ04]Ding, L., Finin, T., Joshi, A., Pan, R., Cost, R.S., Peng, Y., Reddivari, P., Doshi, V.C., and Sachs, J. Swoogle: A Search and Metadata Engine for the Semantic Web. In Proceedings of the Thirteenth ACM Conference on Information and Knowledge Management, November 2004 : 652-659

[GMM03] Guha, R., McCool, R., and Miller, E. Semantic Search. In International World Wide Web Conference Proceedings of the twelfth international conference on World Wide Web. ACM Press. Budapest, Hungary. 2003:700-709

[JP03] Janecek, P. and Pu, P. Searching with Semantics: An Interactive Visualization Technique for Exploring an Annotated Image Collection. OTM Workshops 2003: 185-196

[NSD01] Noy, N.F., Sintek, M., Decker, S., Crubzy, M., Fergerson, R.W., Musen, M., Creating Semantic Web Contents with Protege-2000. IEEE Intelligent Systems 48(2): 60-71, 2001.

[Sea03]Seaborne, A. RDQL-A query language for RDF. http://www.w3.org/Submission/2004/SUBM-RDQL-20040109/, 2003.

[SWY75]Salton, G., Wong, A. and Yang, C. S. A Vector Space Model for Automatic Indexing. Commun. ACM 18(11): 613-620 (1975)

Context-Driven Information Demand Analysis in Information Logistics and Decision Support Practices

Magnus Lundqvist[1], Kurt Sandkuhl[1], Tatiana Levashova[2], Alexander Smirnov[2]

[1]School of Engineering at Jönköping University,
Gjuterigatan 5, SE-551 11 Jönköping, Sweden
{magnus.lundqvist, kurt.sandkuhl}@ing.hj.se
[2]St.Petersburg Institute for Informatics and Automation of the Russian Academy of Sciences,
39, 14th line, St.Petersburg, 199178, Russia
{oleg, smir}@mail.iias.spb.su

Abstract

Decision making and knowledge intensive work requires information supply tailored to the need of the user in question. Context management is considered a key contribution to this objective. The paper investigates context definitions and representations from two different viewpoints: information demand analysis and decision support. Discussion and comparison of these viewpoints includes enterprise models as a source for information demand analysis, ontologies and object-oriented constraint networks as representation technique and correspondences between information demand context and context in decision support.

Introduction

Making decisions, solving problems, and performing knowledge intensive work require readily available information. Today, many different approaches exist to provide users with streamlined information and knowledge to better support them in the process of decision making and problem solving. Among these approaches are context-based decision support and problem solving (Smirnov et al. 2005) and Information Logistics, a demand-driven approach to information supply (Deiters et al. 2003).

The context-based approach to decision support focuses on dynamic problem modeling and solving for decision support. It involves integration of knowledge represented by multiple domain ontologies into context sensitive knowledge. Context sensitive means (a) that knowledge relevant to a problem at hand or situation is integrated, and (b) the integrated knowledge is linked to information sources providing up-to-date information. Main technologies supporting this approach are *ontology management*, *context management*, and *constraint satisfaction*.

While context-based decision support focuses on providing information necessary to solve problems Information Logistics has a similar but somewhat wider perspective on providing users with information. From an Information Logistics point of view only information considered relevant with respect to such aspects as time, location, organizational role, and work activities should be provided to the users. Development of methods, tools and techniques for the analysis of information demand (ID) is the core of the information demand based approach to Information Logistics.

The paper is devoted to a study of context models used within information demand analysis and decision support approaches. The first part of the paper will introduce the information demand viewpoint including relevant definitions and the role of enterprise modeling. The second part focuses on use of context in decision support, which encompasses representation means and decision making stages. The third part finally compares the two viewpoints and draws conclusions.

Context as Dimension of Information Demand

Ongoing research at Jönköping University aims at developing methods, tools and techniques for analyzing ID and to develop systems providing demand-driven information supply. The starting point for work in this area is the following definition of ID:

> Information Demand is the constantly changing need for current, accurate, and integrated information to support (business) activities, when ever and where ever it is needed.

Among other implications, this definition implies that

- models representing ID need to be able to capture the dynamics of information demand in order to reflect changes over time,
- the context, in which the demand exists, as well as some mechanism for understanding when a switch in context takes place has to be provided.

The above clearly identifies the complexity of ID as a concept. It has been proposed that this complexity can be handled by breaking down the concept into several different but interconnected dimensions as *Context*,

Situation, *Information Demand*, and *Plan/Agenda* (Lundqvist & Sandkuhl 2004).

To be able to support activities and provide integrated information it' is necessary to capture and evaluate information about these activities. Thus the concept of *Context* is considered to be the most important aspect of ID defining the settings in which the users' ID exists.

Many different definitions of context exist in areas such as ubiquitous computing, contextual information retrieval, etc. but for the purpose of information demand analysis, context is here simply defined as:

> An Information Demand Context is the formalized representation of information about the setting in which information demands exist and is comprised of the organizational role of the party having the demand, work activities related, and any resources and informal information exchange channels available, to that role.

In this definition, several important concepts can be identified, the central of them being the *Role*. Thus, when *Context* is mentioned here it is considered to be the context of a particular role. It could be argued that it is equally relevant to speak of an ID as related to a specific activity and that some resources are necessary to perform some activities no matter who performs them but *Role* is nevertheless the one concept that in a natural way interconnects the others.

In its simplest form *Role* can be described as a part of a larger organization structure clearly defined by it's responsibility within that structure. Associated with that role are a number of activities that fall within, as well as define, the responsibility of that role. Furthermore there are a number of different resources available to a specific role that can be utilized to perform particular activities. Such resources might be anything from information systems to devices or machines used in the activities. Not all resources are available to all roles and not all resources are suitable for use in all activities, hence the connection between *Role*, *Activities*, and *Available Resources*.

Finally it is proposed that an information demand context incorporates *Contact Networks,* which describe the informal information exchange channels that exist between peers despite not being based on, or formally represented in, any organizational structures, process descriptions, or flow charts. Such informal structures might be based on anything from personal networks, the comfort of turning to other individuals with whom a common interests or the same educational or demographical background is shared. These structures are brought into an organization by individuals but connected to a role since the model above does not include humans but rather the formalized view on humans as roles within an organization.

Deriving Information Demand Context

Contextual information can be derived in many different ways and from many different sources, such as interviews with different roles within an organization, work- or information flow analysis, or various kinds of process modeling methodologies. When trying to reduce the effort necessary, there are mainly two ways to do this; (1) by utilizing already existing knowledge about the object being analyzed and (2) generalize and reuse knowledge gained from the analysis of similar objects. One particularly suitable source for deriving large parts of the contextual information necessary, and thereby utilizing existing knowledge, are Enterprise Models (EM).

Enterprise Modeling has been described as the art of externalizing enterprise knowledge. This is usually done with the intention to either add some value to an enterprise or share some need by making models of the structure, behavior and organization of that enterprise (Verndat 2002). The motivation often given to why enterprises should be analyzed and modeled is that it supports management, coordination and integration of such diverse things as markets, processes, different development and manufacturing sites, components, applications/systems, and so on as well as contributes to an increased flexibility, cleaner and more efficient manufacturing etc. Such models usually include (Verndat 2002), business processes, technical resources, information flow, organizational structures, and human resources. If business processes are considered to be sequences of *Activities*, technical resources and information to be *Resources*, organization to be the structure in which roles (humans) can be identified, these model elements also correspond well to the different aspects of information demand context presented above.

Context-Based Decision Support

The approach to context-based decision support aims at modeling the user's (decision makers', and other participants' involved in the decision making process) problem and solving it. The concept "problem" is used for either a problem at hand to be solved or a current situation to be described. The problem is modeled by two types of context: abstract and operational. *Abstract context* is a knowledge-based model integrating information and knowledge relevant to the problem. As knowledge representation means ontology model is chosen. *Operational context* is an instantiation of the abstract context with data provided by information sources.

Decision making is a complex process where a large number of factors can have an effect on a single problem. To naturally take into account the various factors and constraints imposed by the environment, the mechanism of object-oriented constraint networks (OOCN) (Smirnov et al. 2003) is employed. The problem is modeled by a set of constraints. Constraints can provide the expressive power of the full first-order logics (Bowen 2003) that is tended to be used as the key logics for ontology formalization.

Problem solving within the model of decision making process (Simon 1987) suggests resolving a problem by selecting a "satisfactory" solution. Within the approach the

problem expressed by a set of constraints is to be solved by a constraint solver as *constraint satisfaction problem* (CSP). A result of CSP solving is one or more satisfactory solutions for the problem modeled.

CSP model consists of three parts: a set of variables; a set of possible values for each variable (its domain); and a set of constraints restricting the values that the variables can simultaneously take. To express the problem by a set of constraints that would be compatible with ontology model and with internal solver representations, the formalism of OOCN is used. Typical ontology modeling primitives are classes, relations, functions, and axioms. A correspondence between the primitives of the ontology model and OOCN is shown in Table 1.

Table 1. Primitives of Ontology Model and OOCN

Ontology Model	OOCN
Class	Object
Attribute	Variable
Attribute domain (range)	Domain
Axioms and relations	Constraints

Decision support within the approach is considered consisting of two stages: a preliminary stage and a decision making stage. The *preliminary stage* is responsible for preparedness of a decision support system to make decisions. Activities carried out at this stage are:

- Creation of semantic models for components of a decision support system. The following components are defined: *information sources*, *domain knowledge*, and *users*. These components are modeled as follows: *domain knowledge* is modeled by ontology model; semantics of *information sources* is described by *information source capabilities* model; *users* are modeled by *user profile* model. All the components are represented by OOCN formalism;
- Accumulation of domain knowledge. The approach relies on an availability of sufficient domain knowledge represented by multiple ontologies using the internal representation. As a repository for the collected knowledge an ontology library serves. The domain knowledge is collected before it can be used in problem solving and decision making. Knowledge collecting includes phases of knowledge representation and integration;
- Coupling domain knowledge with information sources. In order to obtain up-to-date information from the environment, ontologies are linked to information sources (sensors, Web-sites, databases, etc.) that keep track of environment changes. Applying the internal representation, attributes of domain ontology and attributes of the representations for information sources and users are linked by associative relationships.

The *decision making stage* concerns integration of information and knowledge relevant to the problem, problem modeling and solving. The starting point for this stage is the user request containing a formulation of the user problem in a user presented view. Based on the results of request recognition the knowledge relevant to it is searched for within the collected knowledge, extracted, and integrated. Ontology-driven knowledge integration enables to involve methods of consistency checking for the integrated knowledge. To operate on the extraction of relevant knowledge, its integration and consistency checking *ontology management* techniques are used.

The consistent knowledge is considered as an *abstract context* that is an ontology-based problem model supplied with links to information sources that will provide data needed for the given problem. The linked information sources instantiate the abstract context producing the *operational context* that is the problem model along with problem data. Obtaining information, its organization in contexts, and context versioning are *context management* issues.

Context-Based Decision Support for Information Demand Definition

In order to compare the two different viewpoints on context presented, dimensions of information demand combining the two viewpoints have been identified.

The user plays different roles depending on the activities this user carries out and other factors. E.g., within a decision support system the user plays the role of a decision maker and, as well, can have several more roles within the organization.

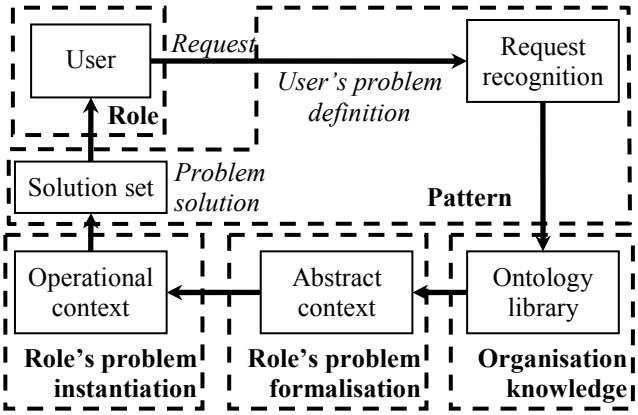


Figure 1. Dimensions of information demand within decision support

The ontology library can include ontologies representing organization knowledge that can be considered to be domain knowledge. Organization knowledge is made up of knowledge about organization structure, activities carried out by the organization, resources deployed, an organization strategy, etc. This knowledge captured by the abstract and operational contexts reports what part of organization knowledge is useful in the context of a given problem. Accumulation of knowledge in the library allows transforming tacit knowledge into explicit one through its formalization.

The abstract context modeling the user's problem can be considered as a formalized model of the problem defined by a certain role. From the information demand perspective an abstract context states information demand as need for integrated information to solve the user problem.

Consequently, the operational context being an instantiation of the problem model describes information demand as constantly changing need for current and accurate information.

The user requests along with the set of problem solutions are considered as a step towards derivation of reoccurring patterns of information demand. The user request formulates the problem at hand under certain settings and conditions. The solution set contains problem solutions for the given problem situation. An analysis of the requests and respective solution sets is supposed to allow deriving common problems and a general solution for a particular problem or problem type. Furthermore, the analysis is thought to be helpful in the identification of types of problems intrinsic to the particular role.

By accepting Enterprise Models as a source for capture of information demand and contextual information derivation, such models is considered to provide with beneficial ideas and knowledge for the decision support in a business application.

Within the decision support approach users are modeled by user profiles. The structure of a profile provides structure elements for storage and accumulation of information characterizing the user and his/her activities. Some of these elements, e.g. organizational belonging, activity, etc. can be adopted from the Enterprise Models.

Knowledge in the Enterprise Models is based on is believed to serve as domain knowledge to be included in the ontology library. Different Enterprise Models can provide different viewpoints on the enterprise knowledge; additionally, they can supplement knowledge of each other.

Conclusions

In order to simplify and shorten the process of developing Information Logistical- and context-based decision support applications as well as ensuring the quality and usefulness of such applications the underlying information demands of the users need to be known. It has been suggested that such demands can be identified and realized through the understanding of the setting in which they exist, i.e. the context in which a user has a particular problem. It has also been shown that one possible source for deriving information demands are Enterprise Models that externalize existing knowledge about an enterprise and thereby shorten and simplify the process of identifying such demands.

The study of context models used within the information demand analysis and decision support approaches facilitate the identification and representation of information demand dimensions intended to be used in the identification of information demand patterns.

The context model used in the decision support approach reflects the needs for integrated information to solve the user problem and for up-to-date and accurate information. The context model of the approach to information demand analysis is believed to be beneficial to the decision support approach in the way it takes into account information that is sensitive to the user role, not only relevant to the user problem.

The two approaches are mutually beneficial in information and knowledge formalization. The decision support approach relying on formalized information and knowledge is a step toward formalization of personal and tacit knowledge taking place while informal information exchange. This formalized knowledge can enrich the ontology library with new knowledge. The explicit formalization of the user's problem within the decision support approach is supposed to enable the identification of problem types intrinsic to the particular role as well as facilitate the derivation of information demand patterns.

Acknowledgement

Parts of the research were done within the projects "Knowledge Supply for Regional and Inter-Regional SME-Networks" funded by the Swedish Foundation for International Cooperation in Research (STINT), 2003-2005; # 16.2.35 and # 1.9 of research programs of RAS; and grant # 05-01-00151 of RFBR.

References

Bowen, J. 2003. Constraint Processing Offers Improved Expressiveness and Inference for Interactive Expert Systems. In *International Workshop on Constraint Solving and Constraint Logic Programming*, LNCS, 2627:93–108, Springer, 2003.

Deiters, W.; Löffeler, T.; Pfenningschmidt, S. 2003. The Information Logistical Approach Toward a User Demand-driven Information Supply. In D. Spinellis, ed., *Cross-Media Service Delivery*, 37—48. Kluwer Academic Publisher.

Lundqvist, M. and Sandkuhl, K. 2004. Modeling Information Demand for Collaborative Engineering. In *Proc. 2nd Intl. Workshop on Challenges in Collaborative Engineering*, 111-120. VEDA, Slovak Academy of Sciences.

Simon, H. A. 1987. Making management decisions: The role of intuition and emotion. *Academy of Management Executive*, 1:57—64.

Smirnov, A.; Pashkin, M.; Chilov, N.; Levashova, T. 2005. Ontology–Based Knowledge Repository Support for Healthgrids. In *From Grid to Healthgrid, Proceedings Healthgrid 2005,* Solomonides T.et al.(eds.), 47—56. IOS Press.

Smirnov, A.; Pashkin, M.; Chilov, N.; Levashova, T. 2003. KSNet-Approach to Knowledge Fusion from Distributed Sources. *Computing and Informatics*, 22:105—142.

Verndat, F. B. 2002. Enterprise Modeling and Integration (EMI): Current Status and Research Perspectives. *Annual Reviews in Control* 26:15—25.

Legal Ontology of Contract Formation:

Application to eCommerce

John W. Bagby and Tracy Mullen

School of Information Sciences and Technology
The Pennsylvania State University, University Park PA U.S.A.
{jbagby, tmullen}@ist.psu.edu

Abstract

Artificial intelligence (AI) has diffused slowly into law, regulation and public policy. The development of complex, reasoning-based applications may be impeded by the structure of legal knowledge that is unlike many other learned professions and scientific domains, law is completely dependant on natural language for the identification of salient factors and determining principles making it difficult to construct necessary or sufficient conditions to produce definite outcomes. Further AI developments in law, regulation and public policy may require much more concentrated effort in representing legal rules, case interpretations and practitioner insights into ontologies. This paper describes our ongoing conversion of previously existing expert systems governing contract formation derived from the Uniform Commercial Code, while integrating electronic contract formation under the Uniform Electronic Transactions Act, into a knowledge-based system using the Web Ontology Language OWL.

Introduction

Artificial intelligence (AI) has diffused slowly into law, regulation and public policy. This research recognizes that AI is inherently interdisciplinary in the law domain requiring domain expertise in both AI and law. Useful expertise in law may come from legal experts and from process observation.

Statutes are legislation embodied in codes, such as the Uniform Commercial Code (UCC) discussed here. Codes may be ideal AI and ontology focci because adapting them to relationship linking and rules-based encoding is transparent. AI cannot rely solely on encoding formal code structure because such efforts are incomplete. They miss practitioner expertise dependant on key relationships and decision heuristics of practitioners and process experts from sociology, political economics, logistics and operations research. The UCC is modeled here because it represents a useful middle ground that overcomes the limitations of reliance on formal code. The UCC is essentially a composite of experience and formalism. The UCC has a unique heritage, derived from the Law Merchant and Lex Mercatoria, essentially codifications of actual practice. The UCC is not primarily a normative codes drafted by inexperienced legislators. AI work on real-world statutory codes like the UCC has both the coding advantages of statutes but is enlightened by realistic experience from development in real-world settings.

This paper describes conversion of an expert system on law from the UCC that covering contract formation, specifically the "Battle of the Forms" problem that resolves the mismatches between contract terms in written offers, acceptances and confirmation. The expert system is being represented in a knowledge-based system using the Web Ontology Language OWL with Jess as inference engine.

Successful Legal Ontology Development

Despite difficulties there have been several interesting experiments in legal AI including some notable functional systems. Consider the complex but deterministic, rules-based systems in commercial tax preparation. Good progress has been made in user assistance from proprietary legal research databases such as Lexis and Westlaw in leveraging traditional legal ontologies. There are numerous electronic transaction processing systems in government found all over the world that assist citizens and regulated entities using AI technologies in areas such as licensing and intellectual property (IP) rights.

New services developed by legal research databases are good predictors of successful AI and ontology work in law because such profit-seeking information services likely invest in AI innovation where a reliable cash flow is predicted. Proprietary databases automate and enhance traditional strategies using key word in context search and retrieval, natural language queries, relevance prioritization with reliability measures, and easy resumption of prior query direction. Recent AI advances permit users easy access to context and subject-sensitive information.

AI Challenges in the Law Domain

The structure of legal knowledge inhibits more complex, reasoning-based AI applications. Many learned professions

and scientific domains differ from law, which is not generally derived from empirical research. Law is an open textured domain requiring AI techniques to classify, link and automate reasoning. Further developments in legal AI may require concentrated effort that starts with the formal statutory structure of legal rules, then modifies with case interpretations and practitioner insights into ontologies.

As in some other professional domains, AI in law runs a malpractice liability risk. AI inference holds promise to improve practitioner reasoning, particularly from the exhaustive search capability. AI in law will likely remain imperfect as sufficient and complete substitute for experienced professional practitioners [Hassett]. Lamkin hypothesizes that legal liability may befall owners or operators of expert systems in medicine if there are misdiagnosis or treatment errors [Lamkin]. There is no good reason to distinguish law from the medical context if a liability shield for AI systems is necessary. Judge and Professor Posner suggests the difficulties of any AI system in predicting legal outcomes beyond the role as assistants useful in organizing and seeking information. Posner notes there are many sources for expertise needed for the inference process. [Posner]

Existing legal AI experiments recognize that legal knowledge is first derived from formal law in constitutions, statutes and regulations. Next it must be interpreted in actual cases as precedents. Finally, this doctrinal legal research must be interpreted through experience of various domain experts. Complexity is increased because law differs between states and among nations. Legal advice based on legal research draws upon huge collections of statutes, legislative history, regulations and cases issued by thousands of discrete authorities. All these sources are raw data that require expert interpretation before constituting reliable advice. With law broken down into manageable-sized sub-domains it becomes more susceptible to internal consistency and coherence and less effected by external domains. Consider Groothuis postulate that expert systems in law should provide advice and decision support for more manageable sub-domains such as government-administered social insurance in the Netherlands [Groothuis]. Also consider the decision support expert system in New York that assists prosecutorial choice of cases to investigate and prosecute [Hassett]. Another example is the assessment of evidence in litigation by Levitt [Levitt, et. al.] .

Legal Ontologies Reflecting both Formal Rules and Actual Practice

AI and ontologies in law hold the strongest promise assisting in legal research and inform legal reasoning with quality control as the major objective. Rissland argues that "AI focuses a spotlight on issues of knowledge and process to a degree not found in non-computational approaches." [Rissland] Aikenhead argues that "It is obviously a prerequisite to know what the nature of law is and what the process of legal reasoning involved before incorporating legal knowledge in a computer and making the computer manipulate that knowledge to emulate the legal reasoning process, i.e., the results achieved by lawyers." [Aikenhead] It follows that ontologies are robust when they enrich the deterministic structure of statutory law. Governing statutory codes are the starting places for much AI work. The enhancement of formal law requires two additional levels of domain knowledge. First, case law interpretations add detail and require expert interpretation. Second, heuristics of seasoned practitioners, regulators, litigators, judges, legislators, sociologists, political economists and others are usually relevant heuristics. Consider how Aoki et. al. enhanced an existing general ontology with a case ontology automatically constructed from precedents using international law governed by the Vienna Convention on the International Sales of Goods (CISG) [Aoki, et. al.].

Commercial Law Blending Formal Specificity with Compilations of Reliable Experience

It is unfortunate that very few codes statutes are drafted to facilitate search, analysis or modification by computer. There are clear design benefits for domain with modular organization.. Nevertheless, Blackwel believes there are benefits in object-oriented analysis and design in AI when the domain involves "complex relationships among distinct concepts. [This] structure will allow close consistency with both the real-world situations addressed, and the legal principles applied, by the statute." [Blackwel] Still, there are some better organized codes that transcend a hodge-podge, historical accumulation of political compromises. For example, the Law Merchant and the UCC are models that improve the potential for adaptation through ontologies into AI. First, the UCC is a well-organized code derived from best practice experience accumulated over centuries of commercial conduct making it a codification of practice. The UCC bridges the gap between legislatively prescribed conduct and actual behavior. Ontology based on the UCC are inherently more robust because many details from experience are included. Second, the UCC is organized in modular form enabling analysis and ontological representation. The CISG is derived from the UCC so it promises similar benefits. This research addresses the "battle of the forms" problem in which commercial contract counter-parties construct self-serving documents that usually diverge with at least some terms in disagreement. The UCC §2-207 provides a regime for resolving these disparities reflecting common practice codified as formal law and is adjusted by practitioner heuristics.

UCC Domain Ontology

Ontologies provide an explicit representation of and semantics for domain concepts and properties. This allows for more natural collaboration between humans and computer, and for intelligent automation by software

agents [Berners-Lee]. In the legal domain, there are two different, but complimentary, ontology modeling approaches. The first approach can be characterized as building a "lawyer's ontology". Kabilan and Johannesson's ontology [Kabilan et. al.] draws from international contract law, and represents its conceptual model using the Unified Modeling Language (UML) [UML]. This UML representation can be transformed into various semantic web ontology languages. The second approach follows a "law in practice" or process-based approach based on actual practice for representing legal contracts. The MIT Process Handbook provides the actual business process knowledge used by SweetDeal [Grosof et. al.] encoded in semantic web languages such as DAML+OIL [DAML+OIL] and RuleML [RuleML]. One of their goals is to allow intelligent software agents to play a larger role in automating, creating, assessing, negotiating and performing such contracts.

The day to day practice of law combines existing law, practical experience, and various cultural, political, and economic factors. When new situations arise, such as electronic commerce, the law must be updated both by extending it in a coherent manner and through a case-by-case learning of new practices. The U.S.'s UCC is just such a hybrid model that we hope will allow us to build a composite "lawyer's ontology" that has been refined with law from actual practice.

An existing expert system on contract formation under UCC [Bagby], see Figure 1, provides us with our initial framework for ontology. Our focus area is the "Battle of the Forms" (UCC 2-207), which defines when mismatches between contract terms still allow for a legal contract to exist. Since the original expert system was intended to be used by lawyers who understand basic domain concepts, our first step in transforming this system into a knowledge-based system requires incorporating de jure formal terms and rules from UCC Article II into the legal ontology. Our eventual goal is to explore how it can be useful for electronic agents to navigate.

The ontology is being built in Protégé [Noy et. al.] using the OWL Web Ontology Language [OWL]. For each domain concept, we attach a definition from either UCC Article II code, standard textbook or other authority. Thus for Merchant, we have a description paraphrased from UCC 2-104. In the future, we would like to link to a Legal Dictionary such as the European Legal RDF Dictionary [LEXML]. Currently we are defining necessary property relationships, such as the "hasSpecialDuties" property of merchants, shown in Figure 2. This property helps capture that UCC Battle of the Forms assumes that merchants can assume additional duties to make contract formation more flexible that non-Merchants should not have to assume. Figure 3 shows our current prototype UCC ontology. Our next steps are 1) to further define properties and property restrictions, and 2) to incorporate Jess (via JessTab in Protégé [Eriksson]) to reason over individual contracts. In step 2, we will start by essentially recreating the existing expert system as an information retrieval system, but defining the rules based on the underlying ontology terms rather than on human understanding.

Conclusion and Future Research

In this paper, we describe our rationale for selecting the UCC commercial laws as the basis for developing a contract formation legal ontology. We describe our initial work on creating a legal ontology for this domain. The authors plan on extending this work to consider several sources of electronic commerce laws that have been implemented in the European Union and the United States. For example, the EU Directive in Electronic Commerce (Dir 2000/31/EC) and the Uniform Electronic Transactions Act (UETA) in the United States are developing sufficient rigor to deserve attention, particularly given their focus on automated transactions, concluded by electronic means including electronic agent activities. Follow-on work will address the impact of deploying intelligent software agents as full-fledged legal persons engaged in these types of transactions. This future work will perform exploratory modeling of additional parts of UETA that acknowledge the validity of electronic agent usage and thereby address some of the barriers to e-commerce presented by traditional rules.

References

Aikenhead, Michael, *The Uses And Abuses Of Neural Networks In Law*12 Santa Clara Computer & High Tech. L.J. 31, February 1996.

Aoki, Chizuru, Masaki Kurematsu & Takahira Yamaguchi, *LODE : A Legal Ontology Development Environment* (1998).

Bagby, John, *Artificial Intelligence in Sales Law: A Decision Analysis Approach To Commercial Transactions*, Working paper in Center for Research, College of Business, The Pennsylvania State University, 87-1, March 1987.

Berners-Lee, T., Hendler, J., and Lassila, O., *The Semantic Web*, Scientific American, May 2001.

Blackwel, Thomas F., *Finally Adding Method To Madness: N1 Applying Principles Of Object-Oriented Analysis And Design To Legislative Drafting* 3 N.Y.U. J. Legis. & Pub. Pol'y 227 (1999 / 2000

DAML+OIL, http://www.w3.org/TR/daml+oil-reference/.

Eriksson, Henrik, *Using JessTab to Integrate Protégé and Jess*, IEEE Intelligent Systems, March/April 2003, pp 43-50.

Grosof, B. N., and T. Poon, *SweetDeal: Representing Agent Contracts with Exceptions using XML Rules, Ontologies, and Process Descriptions*, WWW 2003, Budapest, Hungary, May 20-24, 2003.

Groothuis, Marga M. *Expert Systems In The Field Of General Assistance: An Investigation Into Juridical Quality*, 52 Syracuse L. Rev. 1269 (2002).

Hassett, Patricia, *Essay: Technology Time Capsule: What Does The Future Hold?* 50 Syracuse L. Rev. 1223, 2000.

Kabilan, V., and P. Johannesson, *Semantic Representation of Contract Knowledge using Multi tier Ontology*, SWDB 2003, pp 395-414.

Lamkin, Brian H., *Comments: Medical Expert Systems And Publisher Liability: A Cross-Contextual Analysis*, 43 Emory L.J. 731, Spring, 1994.

Levitt, Tod S. & Kathryn Blackmond Laskey, *Symposium: From Theory To Practice: "Intelligent" Procedures For Drawing Inferences In Static And Dynamic Legal Environments: Computational Inference For Evidential Reasoning In Support Of Judicial Proof*, 22 Cardozo L. Rev. 1691(July, 2001).

LEXML, http://www.lexml.de/rdf.htm.

Noy, Natalya F., Michael Sintek, Stefan Decker, Monica Crubezy, Ray W. Fergerson, and Mark A. Musen, *Creating Semantic Web Contents with Protégé-2000*, IEEE Intelligence Systems, March/April 2001, pp 60-71.

OWL, http://www.w3/2001/sw/webont.

Posner, Richard A. *The Jurisprudence Of Skepticism*, 86 Mich. L. Rev. 827, April, 1988.

Rissland, Edwina L. *COMMENT: Artificial Intelligence and Law: Stepping Stones to a Model of Legal Reasoning*, 99 Yale L.J. 1957 (June, 1990).

RuleML, http://www.ruleml.org.

UML, http://www.uml.org.

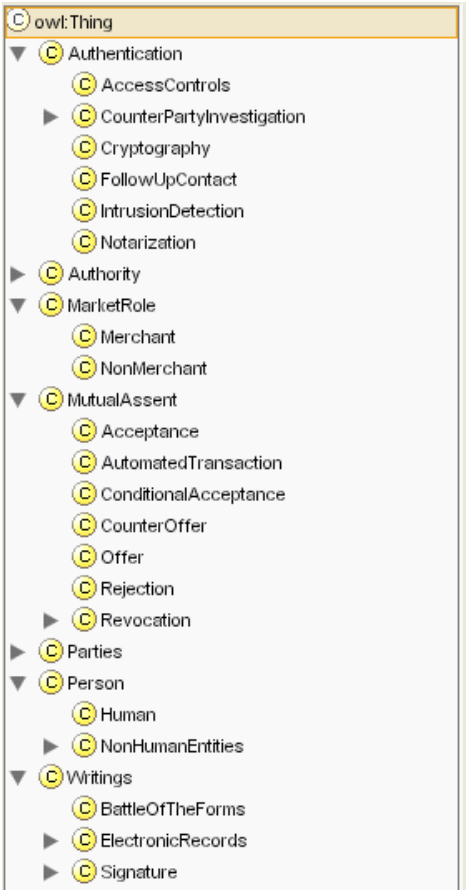


Figure 3: Prototype UCC-based Contract Formation Ontology

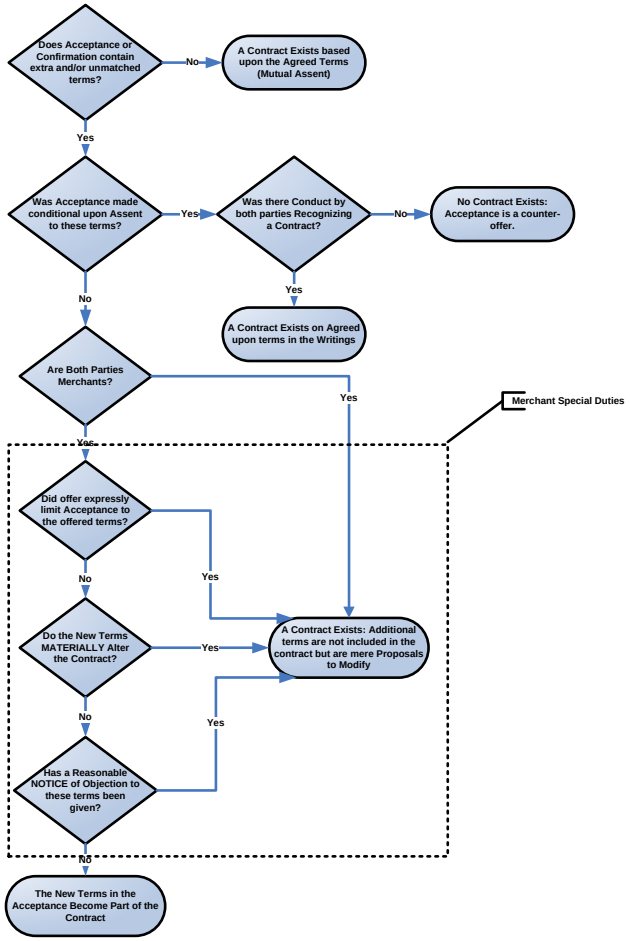


Figure 1: Original Battle of the Forms expert system

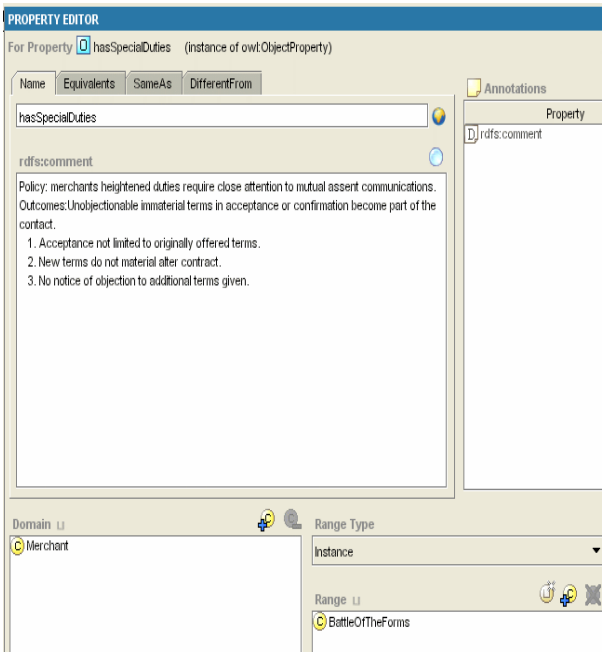


Figure 2: Merchant property of “having special duties”

Context and Ontologies: Contextual Indexing of Ontological Expressions

Leo Obrst, Deborah Nichols
The MITRE Corporation
7515 Colshire Drive, M/S H305, McLean, VA 2210
{lobrst, dlnichols}@mitre.org

Abstract

This paper discusses aspects of context as applied to ontologies. In particular, we note some formalizations of context that have been applied to ontologies such as Menzel (1999) and Akman & Surov (1996, 1997), that have largely been framed in terms of theories such as Situation Theory (Barwise & Perry, 1983) which originated in natural language semantics. We also mention the notion of labeled deduction (Gabbay, 1996) and speculate on its prospective use in the contextualizing of ontologies. The latter can be viewed as a mechanism for annotating ontological assertions and proofs with contextual information about provenance, security, strength/confidence of assertion, and aspects of policy. Labeled deduction correlates one or more logics, with one logic addressing the primary assertion or inference step and another logic addressing the label or annotation of that assertion or inference step.

The Need for Contexts for Ontologies

In recent years, ontologies have been proposed as models which represent the common, shared semantics of domains or subject areas (see Guarino (1998), Guarino, Welty, Smith (2001), Guarino, Varzi, Vieu (2004)). Domain-spanning middle and upper ontologies (Semy et al, 2004; IEEE SUO) have also been proposed, the better to situate and align domain ontologies by axiomatizing common semantics shared by nearly every domain, and allowing those domains to inherit the common semantics. The emerging Semantic Web (Daconta, Obrst, Smith, 2003; Berners-Lee et al, 2001) has more recently defined knowledge representation language standards such as RDF/S, OWL, and extensions of these including Semantic Web Rule Language (SWRL) and OWL-FOL, a first-order logic extension of OWL.

An increasingly important issue in the use of ontologies and the Semantic Web is that of context, i.e., 1) how should an ontology be interpreted in specific, changing contexts, and 2) how can ontologies incorporate the notion of context? Contexts here can be considered specific views of domains, dependent on the user, organization, etc., and their needs and intents.

Increasingly, the notion of context with respect to ontology needs to be addressed. Is a context embedded within a given ontology (where the ontology is viewed as a theory or set of logical theories about a domain)? Is a context with respect to an ontology, i.e., with respect to a particular interpretation of a theory or set of theories, and thus outside the ontology as theory, leading us to view a context as encapsulating ontologies and changing the interpretations of those ontologies in this context as opposed to that context? Is a context a *first-class citizen* of the logic of the ontology? Is it a *microtheory* ala Cyc (Blair et al, 1992), meaning a portion of the (monolithic) ontology that is separable from other microtheories, and thus with respect to those possibly containing contradictory assertions? Should hybrid logics and reasoning methods, as for example discussed in Audemard et al (2002) and Giunchiglia et al (2000), be used?

In the Semantic Web, ontologies expressed in OWL, possibly using SWRL and other extensions, have annotations – annotations on the classes, properties, and instances, but also on the ontologies. These annotations can carry information about the construct, possibly its security, version, provenance, strength or confidence of belief, etc. Currently, these annotations are non-symbolic and uninterpreted, in fact, uninterpretable under the current semantics of OWL. Inference engines that work on OWL ontologies can provide whatever interpretation they desire to these annotations. Similarly, reification in RDF is a statement S2 about a statement S1: a triple along the lines of S2: <john, states, S1>. The truth of S2 cannot be determined; there is no semantics for reification in RDF, only a syntax which a given inference engine is free to semantically interpret as it will. This is problematic, insofar as reification in RDF is used to capture belief information, in particular.

The general problem is therefore: if you make statements about statements or annotate statements with statements in ontology languages, should these be semantically interpreted, and if so, how? In general, statements about statements are formally representable only in second-order logic (however, reification in RDF is first-order). Can these annotations also act as context determiners, and if so how? Because these annotations begin to look like indices in a context structure, i.e., guiding the interpretation of the assertion (or inference step) so annotated, how do we deal with them? How might we formalize context and its interaction with the logical assertions of ontologies? We assert in this paper that these annotations indeed create a context for the interpretation of ontologies. Is this the only

notion of context? No, but it may be that the mechanisms for the multiple notions of context are similar or, in fact, the same.

Furthermore, there is overlap here with the evolving notion of *policy*, especially with respect to Semantic Web ontologies. *Policy* we take as really an aspect of formal pragmatics, as opposed to just the base formal semantics, i.e., policy involves how the semantics should be interpreted in a given context, with the policy theory (ontology) ensuring the correct intent of the policy for a given semantic interpretation, and thereby ensuring the correct usage of the given semantics as expressed in the ontology/ies of the site or enterprise that has propounded the policy.

Formalization of Context for Ontologies

Traditional formalizations of context such as McCarthy (1987, 1991, 1993), Guha (1991), McCarthy & Buvač (1997) and the related notion of *microtheory* in Cyc (Blair, et al, 1992; Lenat & Guha, 1990; Lenat, 1998) introduced the notion of *ist(c, p)*, i.e., a proposition *p* is true (ist) in a given context *c*, a so-called lifting axiom (of a proposition's truth value from one context to another). As Menzel (1999) points out, these formalizations, including that of Akman & Surav (1996, 1998), propose a so-called "subjective conception" of context, meaning one which defines contexts as sets of propositions, i.e., as theories related via an entailment relation, and typically as a set of beliefs of aperson or agent – hence, *subjective*. Menzel (1999), however, proposes an "objective conception" of context, a shared context among agents that views the truth of a proposition not as a logical relation (such as entailment) between the proposition of a context and other propositions, but instead as a correspondence relation between the proposition and the world – hence, *objective*.

This "correspondence" relation is interesting in a number of ways, including its apparent correlation to the notion of *compatibility* of contexts developed in the *local model semantics* of Giunchiglia & Ghidini (1998), Giunchiglia & Bouquet (1997, 1998), and related to Obrst et al (1999a-b). In addition, of course, it acts as a refinement of the *accessibility* relation between worlds in possible worlds semantics (and which, however, is usually taken to be an entailment relation), which is why Menzel (1999) proposes the use of Situation Theory (Barwise & Perry, 1983), which explicitly intends to establish more granular formal contexts in natural language semantics than the usual notion of possible worlds, i.e., situations. Situation theory and a similar theory, Discourse Representation Theory (Kamp & Reyle, 1993), attempt to extend the original focus of natural language semantics from the sentence to the discourse level, including the formal pragmatics of language. Stalnaker (1998) is also relevant here.

Recently, there has been research addressing ontologies and contexts with respect to Semantic Web ontology languages such as OWL. In particular, Bouquet et al (2004) build on Giunchiglia & Ghidini (1998), and extend OWL to include contexts, as Context-OWL or C-OWL, in which mappings among ontologies are first class citizens in their own right, represented independently of the ontologies they link.

Contextual Indexing of Ontological Expressions

One prospective accommodation of contexts to ontologies involves the notion of labeled deduction (Gabbay, 1996; Basin et al, 2000). In labeled deduction, multiple logics are correlated. In some natural language processing usage of labeled deduction, the formal syntax of an expression is correlated with its formal semantics (Finger et al, 1997; Kempson, 1996; Moortgat, 1999).

Some examples from formal linguistics may help illustrate how labeled deductive systems (LDS) work. In Figure 1 (from Gabbay & Kempson, 1992; adapted from Kempson, 1996, p. 569), the Modus Ponens (MP) proof structure contains units of the form label-plus-formula, e.g., **α:P,** with α labeling the formula **P** (with **P, Q** ranging over logical types **e, t, e$\rightarrow$t** – roughly, type entity, type truth value, and functional type entity to truth value, respectively). In this example, the conclusion **$\beta(\alpha)$:Q** signifies the function application of β on α in the label of the formula **Q**. In this natural language parsing application (using the Curry-Howard isomorphism of types as formulae), words are labels on their types, and successive Modus Ponens applications build up a semantic interpretation of a sentence via simultaneous function applications on the labels.

$$\frac{\alpha:P \qquad \beta:P \rightarrow Q}{\beta(\alpha):Q}$$

Figure 1. Labelled Deductive System: Modus Ponens

In Figure 2 (Kempson, 1996, p. 574), a rule of $\rightarrow$Introduction is given, where the label builds a

Assume:	α:P … $\beta(\alpha)$:Q
$\lambda x\, \beta(x)$: P $\rightarrow$ Q	

Figure 2. Labelled Deductive System: $\rightarrow$Introduction

λ-abstraction which records where the assumption has been retracted. Such a representation might be used for ellipsis in natural language discourse, where the resulting lambda term can be then be bound to another premise.

In other more typical logical usages, an assertion or inference step is annotated with other logical information, so that multiple logics exist and act over the same expression. For each primary logical assertion or deductive step, annotations exist. These annotations (labels) are themselves symbolically interpreted according to the logic they are expressions of, at each step in the primary assertion or deductive step. Typically, the annotations are expressed in simpler logics than the primary assertion/deductive step. Consider a very simple example, where the label of each formula in the MP proof above is just $\mathbf{t_1}$, designating a specific time at which the formula is true. From $\mathbf{t_1}$:**P**, $\mathbf{t_1}$:$\mathbf{P \rightarrow Q}$, one concludes $\mathbf{t_1}$:**Q**. The effect is therefore that the most computationally resource-intensive deduction using the logical assertions drives the inference, with the annotations (expressing security, strength of belief, provenance information) represented in less expressive and therefore more efficiently executed logics (typically propositional logics, some of which can be implemented in bit-vector operations). The result is that a Modus Ponens proof can simultaneously cause the composition of security and/or belief-confidence annotations according to simpler logics, and propagate the annotations through the ontological space.

Labeled deduction, therefore, may be a mechanism by which contexts expressed as indices representing security, belief, provenance, and other policy (formal pragmatic) determinants may influence the interpretation of ontological (semantic) expressions. For example, Rasga et al (2002) with regard to modal logic, discusses using a labeled formula **x**:φ which means that φ holds at world **x** in the underlying Kripke structure (model), and then defining rules which separately and simultaneously work on the labels and the formulae. Blackburn (1999, 2000) *internalizes* labeled deduction by moving its methods from the (external) metalanguage to the object language (propositional modal logic) by introducing as labels **i** *nominals*, each of which is true at exactly one state in the model. So a formula with a nominal label "**i:** φ will be true at any state in a model iff φ is true at the state that **i** labels" (Blackburn, 2000, p. 137-138). The resulting logic is a hybrid logic with two sorts: propositions and nominals.

In this short paper, we can only suggest the possible use of labeled deduction for contextual indexing of ontological expressions. For example, one might consider a very simple system for a security context, where individual propositions (facts or assertions in an ontology) and ontology rules are labeled with their respective security classifications. The resulting system (using MP) might look as in Figure 3.

α:**P**
β:$\mathbf{P \rightarrow Q}$
———————
$(\alpha * \beta)$:**Q**

where $(\alpha * \beta)$ **is defined as** (α, β **elements of a poset and** $\geq$ **is a partial ordering):**
i. $(\alpha * \beta) = \alpha$ **if** $\alpha \geq \beta$
ii. $(\alpha * \beta) = \beta$ **if** $\beta > \alpha$

Figure 3. LDS MP: Security Labels + Ontology Expressions

Acknowledgments

The views expressed in this paper are those of the authors alone and do not reflect the official policy or position of The MITRE Corporation or any other company or individual.

References

Akman, Varol and Mehmet Surav. 1997. The Use of Situation Theory in Context Modeling, Computational Intelligence 13(3), pp. 427-438, August, 1997.

Akman, Varol and Surav, Mehmet. 1996. Steps toward Formalizing Context. AI Magazine 17(3), pp. 55-72, 1996.

Audemard, Gilles; Piergiorgio Bertoli; Alessandro Cimatti; Artur Kornilowicz; Roberto Sebastiani. 2002. Integrating Boolean and Mathematical Solving: Foundations, Basic Algorithms and Requirements". In "Artificial Intelligence, Automated Reasoning, and Symbolic Computation. Proc. of Joint AISC 2002 and Calculemus 2002." Marseille, France, 2002. LNAI N.2385 © Springer.

Barwise, Jon, and John Perry. 1983. Situations and Attitudes. Cambridge: MIT Press.

Basin, D., M. D'Agostino, D. Gabbay, S. Matthews, and L. Viganò, editors. 2000. *Labelled Deduction*, pages 107-134, Kluwer Academic Publishers, 2000.

Berners-Lee, T; J. Hendler; and O. Lassila. 2001. The Semantic Web. In The Scientific American, May, 2001. http://www.scientificamerican.com/2001/0501issue/0501berners-lee.html.

Blackburn, Patrick. 1999. Internalizing Labelled Deduction. In Proceedings of Hylo'99, First International Workshop on Hybrid Logics. July 13th, 1999, Saarbrücken, Germany. Published in Journal of Logic and Computation, 2000 10(1):137-168.

Blair, Paul; Guha, R.V.; Pratt, Wanda. 1992. Microtheories: An Ontological Engineer's Guide. Technical report Cyc-050-92, March 5, 1992, Cycorps, Austin, TX. http://www.cyc.com/tech-reports/cyc-050-92/cyc-050-92.html

Bouquet, Paolo; Fausto Giunchiglia; Frank Van Harmelen; Luciano Serafini; Heiner Stuckenschmidt. 2003. C-OWL: contextualizing ontologies. "2nd international semantic web conference (ISWC 2003)", edited by Dieter Fensel and Katia p. Sycara and John Mylopoulos, Sanibel Island (Fla.), 20-23 October 2003, pp. 164-179.

Daconta, M., L. Obrst, K. Smith. 2003. The Semantic Web: The Future of XML, Web Services, and Knowledge Management. John Wiley, Inc., June, 2003.
Davey, B.A.; Priestley, H.A. 1991. *Introduction to Lattices and Order*. Cambridge University Press, Cambridge, UK.
Finger, Marcelo; Rodger Kibble; Dov Gabbay; Ruth Kempson. 1997. Parsing Natural Language Using LDS: a Prototype. http://semantics.phil.kcl.ac.uk/ldsnl/papers/fkkg97pnlul.pdf.
Gabbay, Dov. 1996. Labelled Deductive Systems; Principles and Applications. Vol 1: Introduction. Oxford University Press.
Gabbay, Dov; Ruth Kempson. 1992. Natural-language content: a truth-theoretic perspective. In Proceedings of the 8th Amsterdam Formal Semantics Colloquium. Amsterdam: University of Amsterdam.
Giunchiglia, Fausto; Roberto Sebastiani; Paolo Traverso. 2000. Integrating SAT solvers with domain-specific reasoners. Symbolic Computation and Automated Reasoning, St. Andrews, Scotland, UK, August 2000. A.K. Peters Eds.
Giunchiglia, Fausto; Bouquet, Paolo. 1997. Introduction to Contextual Reasoning: An Artificial Intelligence Perspective. Istituto per la Ricerca Scientifica e Tecnologica (IRST), Trento, Italy, Technical report 9705-19, May, 1997.
Giunchiglia, Fausto; Bouquet, Paolo. 1998. A Context-Based Framework for Mental Representation. Istituto per la Ricerca Scientifica e Tecnologica (IRST), Trento, Italy, Technical report 9807-02, July, 1998.
Giunchiglia, Fausto; Ghidini, Chiara. 1998. Local Models Semantics, or Contextual Reasoning = Locality + Compatibility. Principles of Knowledge Representation and Reasoning (KR'98), Proceedings of the Sixth International Conference, Trento, Italy, June 2-5, 1998, Anthony Cohn, Lenhart Schubert, Stuart Shapiro, eds., pp. 282-289.
Guarino, N, ed. 1998. Formal Ontology in Information Systems. Amsterdam.: IOS Press. Proceedings of the First International Conference (FOIS'98), June 6-8, Trento, Italy.
Guarino, N.; Achille Varzi; Laure Vieu. 2004. The Proceedings of the 3rd International Conference on Formal Ontology in Information Systems (FOIS-04), November 4-6, 2004. http://fois2004.di.unito.it/.
Guarino, N.; C. Welty; B. Smith, ed. 2001. The Proceedings of the 2nd International Conference on Formal Ontology in Information Systems (FOIS-01), October 16-19, 2001, Ogunquit, Maine. ACM Press Book Series, Sheridan Publishing, Inc. http://www.fois.org/fois-2001/index.html.
Guha R. V.. Contexts: A Formalization and Some Applications. 1991. PhD thesis, Stanford University, 1991. Also technical report STAN-CS-91-1399-Thesis, and MCC Technical Report Number ACT-CYC-423-91.
IEEE SUO. IEEE Standard Upper Ontology Working Group. http://suo.ieee.org/.
Kamp, Hans; Reyle, Uwe. 1993. From Discourse to Logic: Introduction to Modeltheoretic Semantics of Natural Language, Formal Logic and Representation Theory, Part 1 and 2, Kluwer Academic Publishers, The Netherlands.
Kempson, Ruth. 1996. Semantics, Pragmatics, and Natural-Language Interpretation. In the Handbook of Contemporary Semantic Theory. Shalom Lappin, ed., pp. 561-598. Blackwell.
Lenat, Doug. 1998. The Dimensions of Context-Space. Cycorp, Austin, TX, Technical Report, October 28, 1998.
Lenat, D. and R. Guha. 1990. Building Large Knowledge Based Systems.Reading, Massachusetts: Addison Wesley.
McCarthy, J.; Buvač, Sasa. 1997. Formalizing Context (Expanded Notes). In *Computing Natural Langauge*, A. Aliseda, R. van Glabbeek, & D. Westerståhl, eds., Stanford University. http://www-formal.stanford.edu.
McCarthy, John. 1987. Generality in Artificial Intelligence, Communications of the ACM 30(12), pp. 1030-1035.
McCarthy, John. 1990. Formalizing Common Sense: Papers by John McCarthy. Ablex Publishing Corporation, 355 Chestnut Street, Norwood, NJ 07648, 1990.
McCarthy, John.. 1993. Notes On Formalizing Context. In Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence, 1993.
Menzel, Chris. 1999. The Objective Conception of Context and Its Logic, 1999. Minds and Machines 9(1): 29-56 (Feb 1999).
Moortgat, Michael. 1999. Labelled Deduction in the Composition of Form and Meaning. In Logic, Language and Reasoning. Essays in Honor of Dov Gabbay. U. Reyle and H.J. Ohlbach, eds. Kluwer, 1999.
Obrst, L., G. Whittaker, A. Meng. 1999a. Semantic Context for Object Exchange, AAAI Workshop on Context in AI Applications, Orlando, FL, July 19, 1999.
Obrst, L., G. Whittaker, A. Meng. 1999b. Semantic Interoperability via Context Interpretation, submission, Context-99, Trento, Italy, April, 1999, invited poster session.
Rasga, João; Amílcar Sernadas; Cristina Sernadas; Luca Viganò. 2002. Labelled Deduction over Algebras of Truth Values. In Proceedings of FroCoS'2002, the 4th Workshop on Frontiers of Combining Systems. Springer-Verlag.
Semy, S., M. Pulvermacher, L. Obrst. 2004. Toward the Use of an Upper Ontology for U.S. Government and Military Domains: An Evaluation. MITRE Technical Report 04B0000063, September, 2004.
Stalnaker, Robert. 1998. On the Representation of Context. Journal of Logic Language and Information.

Explaining Question Answering Systems with Contexts

Deborah L. McGuinness

Knowledge Systems, Artificial Intelligence Lab, Stanford University, dlm@ksl.stanford.edu

Abstract

One central issue in the usability of answers is in their understandability. We have focused a line of research on explaining answers from heterogeneous distributed systems with the goal of improving usability and trustability in answers by increasing answer understandability and trustability. We explore ways of providing an interoperable distributed infrastructure that supports explanations containing provenance concerning answers – where information came from, how reliable it is -- along with information about assumptions relied on, information manipulation techniques, etc. The infrastructure utilizes the Proof Markup Language, encoded in the OWL Web Ontology Language so as to be compatible with web standards and also so as to be able to leverage existing web ontologies. In this paper, we expose and explore some issues related to context as it is used in some of our question answering systems.

Introduction

As question answering systems utilize more varied data sources and reasoning methods, it is becoming increasingly important to provide not only answers, but also information about the answers so users (humans and agents), can evaluate and understand the answers. The information may provide details concerning the raw data sources, how recent they are, who authored them, whether they are considered authoritative, etc. The explanations may further contain information about any manipulations that were done to the raw sources – were text extraction techniques used to generate knowledge bases? How accurate are those extraction routines? What reasoning methods and reasoners were used to deduce conclusions? Were the methods heuristic, sound and complete, etc.? Further, were assumptions relied on in the reasoning process? If so, which ones? When this kind of information is available in a machine operational format, for example by being encoded in the Proof Markup Language, then this information can be made available to agents and end users. When explanations contain this kind of meta information about the answer and the answer process, users become empowered to make informed decisions about when to use the answer. The user can access information that can allow them to check how reliable the sources were, how accurate the information manipulation techniques might have been, if assumptions were used that either they agree with or can tolerate, etc. Our approach enables this information to be packaged together in a structured object, sometimes called a "proof object" or "justification object" that can accompany any statement.

Explanation in Context

We propose the use of a portable, distributed infrastructure – Inference Web [McGuinness and Pinheiro da Silva, 2004]— that provides:

- support for registering meta information about objects used in justifications;
- a web compatible, interoperable proof markup language for encoding formal and information justifications [Pinheiro da Silva, McGuinness, & Fikes, 2005];
- services for manipulating proof objects to provide capabilities for browsing, abstracting, checking, and interacting with justifications.

We have used Inference Web to explain answers from applications that use standard first order logic reasoning systems such as Stanford's JTP system, used in applications for example in the KSL Wine Agent Demo [Hsu & McGuinness, 2003] and in DARPA programs such as the Personalized Assistant that Learns program's Cognitive Assistant that Organizes project. It has also been used in a number of other settings that include question answerers that may rely on knowledge bases that use text extraction, such as those from IBM's UIMA effort [Ferrucci & Lally, 2004]. Inference Web has been integrated with UIMA to support the ARDA Novel Intelligence for Massive Data program in a system called KANI so that it can explain answers derived using reasoning over knowledge bases partially generated by text extraction [Welty et. al., 2005].

The KANI system does more than typical theorem proving- it also addresses issues related to typicality and temporal reasoning. It makes decisions using typicality assumptions, such as "People who own businesses typically have offices at the businesses" or "People who have access to a particular telephone line at a particular time may make a call on that line". It needs to then provide explanations of its answers that include information such as which typicality assumptions were

used in the reasoning path. Further, if there is any information about the trust a user may have in the assumptions (or the raw sources or the question answerers, etc., this should also be exposed).

Our explanation infrastructure provides an extensible format that has been adequate for encoding assumptions required for the contextual reasoning required to date. It has also been adequate for encoding the information manipulation steps required to date. It can be viewed as a prototype implementation of a design that supports explanations of hybrid question answering systems that require some support for context. The implementation today in the KANI system does not integrate the trust component [Zaihrayeu et. al., 2005] however a future implementation will include this so that trust levels can be included with explanations. Our interest in this workshop is to gather input both on the existing explanation support for our initial forms of limited context reasoning and to gather additional requirements for explanation support for future, more expansive applications.

Conclusion

Our thesis is that question answering systems may have limited value if they can only answer questions but can not provide explanations along with the answers. We have provided an explanation infrastructure that supports explanations in distributed heterogeneous question answering environments, such as the web, where data sources may have a wide diversity of quality, recency, and reliability. Further, it supports explanations of question answering environments that must take context into account and provide forms on reasoning with context. Our Inference Web framework is being used to support question answering systems in a number of research programs funded by DARPA and ARDA and has been integrated with question answering systems that range from theorem provers, to text analysis engines, to expert systems, to special purpose temporal and context reasoners.

References

David Ferrucci, 2004. Text Analysis as Formal Inference for the Purposes of Uniform Tracing and Explanation Generation. IBM Research Report RC23372.

David Ferrucci and Adam Lally, 2004. UIMA by Example. IBM Systems Journal 43, No. 3, 455-475.

Eric Hsu, and Deborah L. McGuinness. KSL Wine Agent: Semantic Web Testbed Application, Proceedings of the 2003 International Workshop on Description Logics (DL2003). Rome, Italy, September 5-7, 2003. www.ksl.stanford.edu/people/dlm/webont/wineAgent/

Deborah L. McGuinness, 1996. Explaining Reasoning in Description Logics. Ph.D. Thesis, Rutgers University. Technical Report LCSR-TR-277.

Deborah L. McGuinness and Paulo Pinheiro da Silva, 2004. Explaining Answers from the Semantic Web: The Inference Web Approach. Journal of Web Semantics 1(4):397-413.

Paulo Pinheiro da Silva, Deborah L. McGuinness, and Richard Fikes. A Proof Markup Language for Semantic Web Services. Information Systems (to appear)

Paulo Pinheiro da Silva, Deborah L. McGuinness, and Rob McCool, 2003. Knowledge Provenance Infrastructure. In IEEE Data Engineering Bulletin 26(4), 26-32.

Ilya Zaihrayeu, Paulo Pinheiro da Silva, and Deborah L. McGuinness, 2005. IWTrust: Improving User Trust in Answers from the Web. Technical Report DIT-04-086, Informatica e Telecomunicazione, University of Trento, Italy

Christopher Welty, J. William Murdock, Paulo Pinheiro da Silva, Deborah L. McGuinness, David Ferrucci, Richard Fikes. Tracking Information Extraction from Intelligence Documents. In Proceedings of the 2005 International Conference on Intelligence Analysis (IA 2005), McLean, VA, USA, 2-6 May, 2005.

Statement of Interest for AAAI-2005 Workshop on Contexts and Ontologies

Nestor Rychtyckyj

Manufacturing Engineering Systems
Ford Motor Company
Dearborn, MI
nrychtyc@ford.com

Statement of Interest

Since the early 1990s Ford Motor Company Vehicle Operations has utilized a knowledge-based system utilizing description logics to manage the vehicle assembly process planning at Ford's assembly plants. The heart of this system is a knowledge base that contains all of the manufacturing knowledge that is needed to build vehicles at our assembly plants. The following type of knowledge is contained within the system: information about the assembly work, time required to perform an assembly operation, required tooling, part information, a lexicon of terminology that is used by the engineers to describe the assembly build instructions, ergonomic constraints [1] and associated manufacturing information. This knowledge is utilized by Ford engineers and assembly plant operators throughout the world using a global enterprise system known as the Global Process Allocation Study System (GSPAS) [2].

The GSPAS knowledge base is a description logic-based implementation containing knowledge about Ford's vehicle manufacturing domain, including information about tools, parts, assembly instructions, ergonomics issues and lexical information about Standard Language. Standard Language is a controlled language developed at Ford that is used to describe manufacturing instructions in a restricted syntax that is processed by GSPAS. The knowledge within GSPAS is being constantly being updated due to the dynamic nature of the automotive industry [3].

Our current focus is to expand the capability of our systems by tapping into the relevant knowledge that is available throughout the corporation but cannot be easily found or integrated into our existing systems. This type of knowledge includes information about manufacturing best practices, "lessons learned", safety and local plant issues and other related information. All these other knowledge sources cannot be easily integrated due to many factors including missing or incorrect context information, different structure of knowledge bases, mismatched terminology, missing terminology and other related reasons. In many cases we need to deal with unstructured data and text that resides in internal web sites, documents, spreadsheets, existing databases and other knowledge sources.

Figure 1 depicts our current approach to build a structured knowledge representation scheme for manufacturing. The list of knowledge sources that needs to be integrated is listed on the left-hand side under the title of "VOME Knowledge". This contains various sources of knowledge that are relevant for manufacturing. Some of these are already contained in knowledge bases, others may be in databases, spreadsheets or in other formats. Our goal is to develop a structured ontology that will be able to access this knowledge from various sources and make it available for our users at the appropriate time.

Therefore, I am very interested in this workshop to see how different context and ontologies can be combined in terms of integrating information and sharing knowledge across a broad spectrum of application areas.

References

1. Rychtyckyj, Nestor (2004), "Ergonomics Analysis for Vehicle Assembly Using Artificial Intelligence", *Proceedings of the 16th Conference on Innovative Applications of Artificial Intelligence*, July 25-29, 2004, San Jose, CA, pp. 793-798.
2. Rychtyckyj, N., (1999), "DLMS: Ten Years of AI for Vehicle Assembly Process Planning", *AAAI-99/IAAI-99 Proceedings*, Orlando, FL, July 18-22, 1999, pp. 821-828, AAAI Press.
3. Rychtyckyj, N., and Reynolds, R., (2000), "Long-Term Maintenance of Deployed Knowledge Representation Systems", *Proceedings of the Seventh International Conference on the Principles of Knowledge Representation and Reasoning*, April 12-17, 2000, Breckenridge, CO, pp. 494-504, Morgan Kaufmann Publishers.

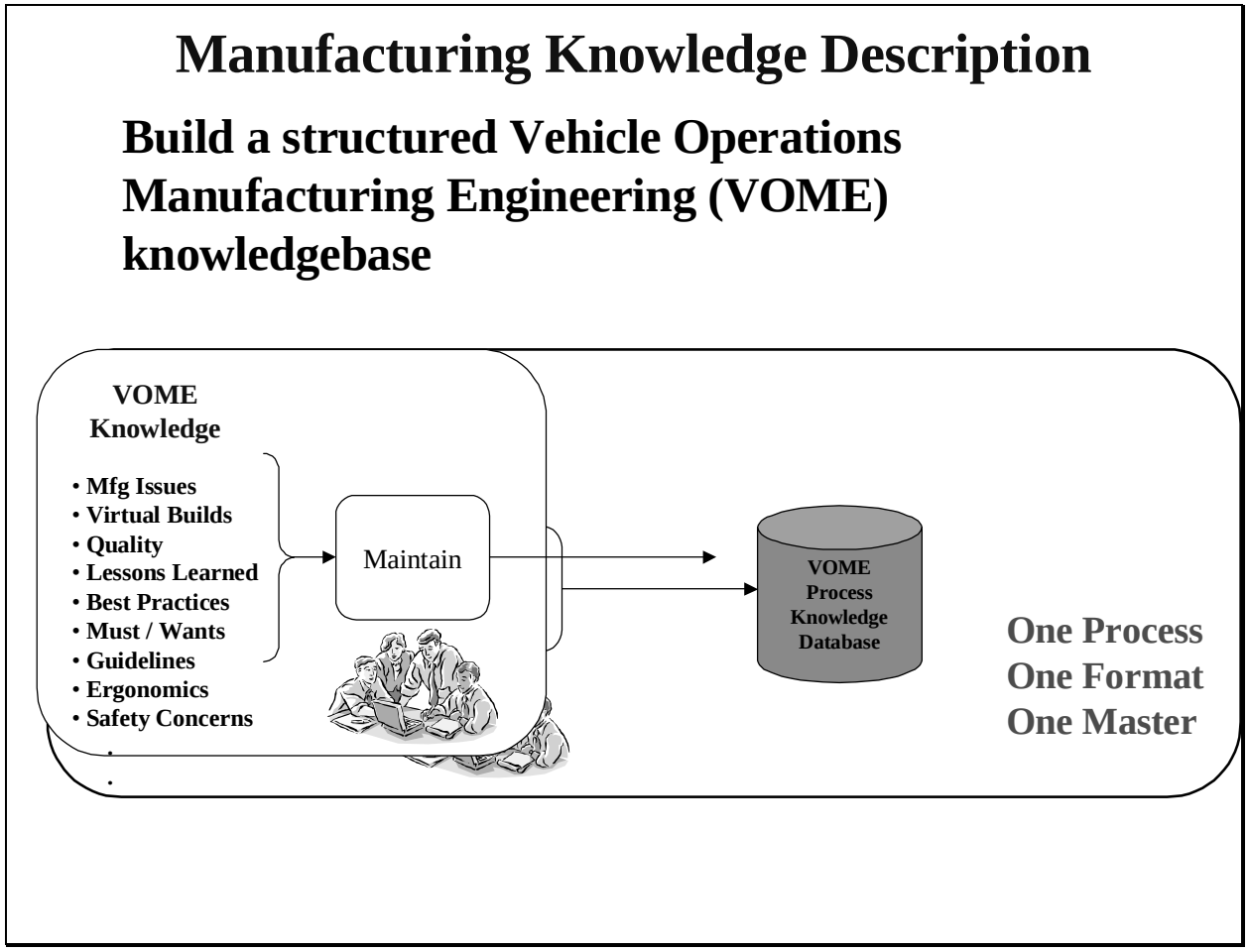


Figure 1: Knowledge Sharing Environment in Manufacturing

Context-aware Policy Matching in Event-driven Architecture

Shao-you Cheng
r93070@csie.ntu.edu.tw

Wan-rong Jih
jih@agents.csie.ntu.edu.tw

Jane Yung-jen Hsu
yjhsu@csie.ntu.edu.tw

Computer Science and Information Engineering,
National Taiwan University, Taiwan

Motivation

Applications for supporting pervasive computing and agility are the current trend in software development. The service-oriented architecture (SOA) enables connection between consumers and service providers in a loosely-coupled way to improve flexibility and extensibility. This architecture is static, which utilizes predefined sequences of actions and fixed policies. Ongoing processes cannot adapt to dynamic changes in the environmental conditions or *context*.

Imagine the situation where a real estate broker shows her client a house for sale, matching the preference profile provided by the potential buyer. While the buyer likes the general location of the house, he considers it unacceptable due to the unexpected traffic noise from a nearby street. Instead of proceeding with the original plan to show another house on the same street, an experienced broker should adjust the plan in light of this additional constraint. The broker connects to the multiple listing service with her mobile device and downloads a newly listed house within minutes of the current location that better satisfies the client's requirements.

Such a scenario can facilitate the introduction of *events* into SOA (He 2003). Real-time changes are modeled as events, which in turn trigger changes of states for the workflow to meet the business needs. In this scenario, events can be "shows client a house", "add client's requirements", etc.

Introduction

This research explores the role of context-aware policy matching in an *event-driven architecture* (EDA). In particular, a context-aware rule engine is adopted to derive conclusions based on the current contexts and business policies. Figure 1 shows the functional modules of the prototype designed to demonstrate the advantages of the proposed approach. In the *Underlying Architecture* layer, standard SOA is combined with EDA. The *Context-Aware Rule Engine* layer consists of three distinct agents for collecting preference profiles, ambient or context information, and dynamic events. All information collected will be forwarded to the Rule Engine, augmented with the Rule Repository and Context Ontology. Results from the Rule Engine will be given to the Action Agent that performs the desired sequence of actions. The following sections contain detailed description of each module.

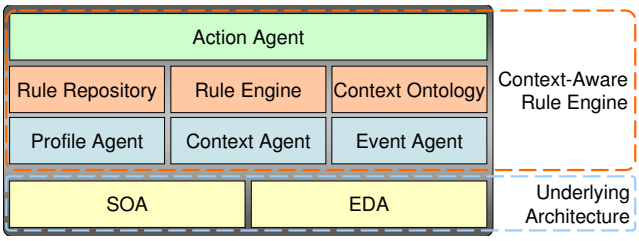


Figure 1: Context-aware policy matching framework

Event-driven Architecture

The SOA use a simple and clear interface to bind all participating software components and provides service reuse. However, it lacks proactive ability. Event notification is the core of EDA, it can tightly bind services, events, and consumers in a dynamic environment. Events are more likely to complement, not replace, services in SOA. Since EDA has been touted as the "the next big thing" on the horizon of software development methodology (Schulte 2004). It is generally believed that event-driven services under SOA gain benefits from the features of both architectures (Hanson 2005).

Conventional event-driven design is also referred to as message-based systems, where a message here is a form of an event. Specifications of WS-Notification family and WS-Eventing define the standard Web services approaches for event handling. When events occur, as shown in Figure 2, service producers publish the messages which will be delivered, e.g. via publish-and-subscribe, to the event consumers.

Context-Aware Rule Engine

The proposed framework utilizes the Jess rule engine (Friedman-Hill 2005) to provide inference results. Jess processes the rules and facts using the Rete algorithm (Forgy 1982). User preferences are represented as policy rules that are matched in the reasoning process.

Contexts refer to the various dynamic aspects of the environment, for example, location, time, and people (Dey

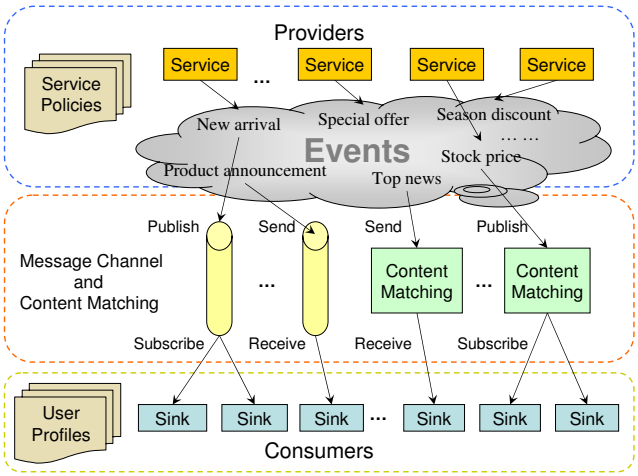


Figure 2: Message-based event-driven architecture

2001). Static rules cannot react to the dynamically changing contexts. As a result, context-aware rules are defined to use dynamic facts in the knowledge base, which may result in different conclusions depending on the current context. In our previous work, we successfully implemented context-aware rule-based reasoning on Java-enabled mobile devices, such as the iPAQ, to perform access control of sensitive information (Jih, Cheng, & Hsu 2005).

A context agent monitors and perceives any environmental changes. Figure 3 shows that context information may be captured by various multi-modal sensors, such as GPS or RFID. The context agent filters the contexts that have been detected, passing the selected contexts to the rule engine. Similarly, the event agent perceives and filters the relevant events of its surrounding area, and the profile agent keeps track of the user's preferences.

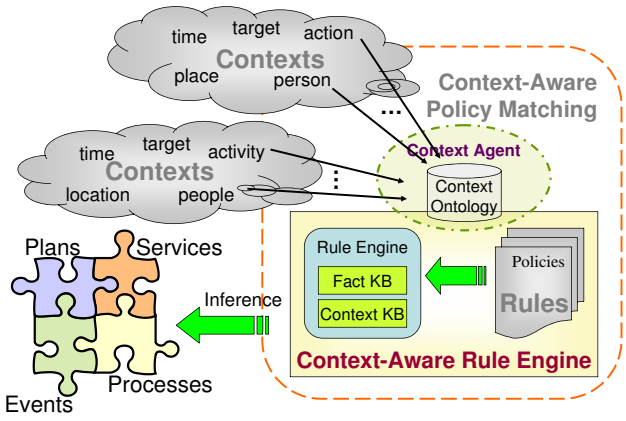


Figure 3: Context-aware rule engine

Now, let's examine the scenario of the real estate broker again. The buyer's rejection of the current house generates two events. First, the buyer is not satisfied with the choice, so he needs additional recommendations. Second, the buyer preference profile is incomplete, and should be modified to reflect his preference for a quiet location. The former event triggers the broker's mobile device to request for additional listing from the server, subject to successful matching between the updated preference profile and updated new listing.

When contexts are encoded in the rule sets, inconsistency in the terminology is not only confusing, but also leads to incorrect reasoning outcomes. Given that most useful contexts are tightly related to the problem domain (e.g. real estate), *ontology* can be used to bridge the semantic gap among different vocabularies. For example, while *location* is a common component of context defined in many applications, sometimes it is referred to as *place*, *space*, or *area*, and so on. A context ontology helps generate a complete model of the different contexts (H.Wang *et al.* 2004) for a given domain. A specific context might directly derive from a more generic one, aggregate to a complex context, or up to an abstract context. Moreover, the context ontology will support hierarchical views of contexts (Gu *et al.* 2004) to improve the reasoning power of the rule engine.

Conclusion

This paper describes the design of a proposed framework for deploying a context-aware rule engine to the event-driven services platform in order to provide agile and real-time services. The design uses an agent architecture and a rule engine for flexibility and scalability in software development. A context ontology is utilized to resolve inconsistent vocabularies in knowledge sharing and rule merging.

References

Dey, A. K. 2001. Understanding and using context. *Personal Ubiquitous Computing* 5(1):4–7.

Forgy, C. 1982. Rete: A fast algorithm for the many patterns/many objects match problem. *Artificial Intelligence* 19(1):17–37.

Friedman-Hill, E. 2005. Jess, the rule engine for the java platform. Sandia National Laboratories.

Gu, T.; Wang, X. H.; Pung, H. K.; and Zhang, D. Q. 2004. An ontology-based context model in intelligent environments. In *Proceedings of Communication Networks and Distributed Systems Modeling and Simulation Conference*, 270–275.

Hanson, J. 2005. Event-driven services in soa. JavaWorld.

He, H. 2003. What is service-oriented architecture? O'Reilly Web services.XML.com.

H.Wang, X.; Gu, T.; Zhang, D. Q.; and Pung, H. K. 2004. Ontology based context modeling and reasoning using owl. In *Proceedings of Workshop on Context Modeling and Reasoning(CoMoRea'04)*, 18–22.

Jih, W. R.; Cheng, S. Y.; and Hsu, J. Y. J. 2005. Context-aware access control on pervasive healthcare. In *EEE'05 Workshop: Mobility, Agents, and Mobile Services (MAM)*.

Schulte, R. 2004. Event-driven architecture: The next big thing. Application Integration and Web Services Summit. Gartner, Inc.

Application of Semantic Web Technologies to UML based Air Force DoDAF Efforts

Dru McCandless, MITRE, mccandless@mitre.org

Introduction

The Department of Defense (DoD) Architecture Framework[1] (DoDAF) "defines a common approach for architecture description, development, presentation, and integration for both operational and business processes. It is intended to ensure that architecture descriptions can be compared and related across organizational and mission area boundaries, including Joint multi-national boundaries and DoD warfighting and business domains."[2] As such, it guides the Air Force's architecture development efforts to describe system(s) performance, interoperability, and processes. A typical architecture consists of several views showing such things as node connectivity, information exchanges, and organizational models. Several UML diagrams, particularly Class, Activity, and Sequence diagrams, are used extensively in DoDAF efforts. However, the end products are intended for a human audience and are not easily machine reasonable.

A natural set of questions are: how can the information contained within these UML models be used to support automated reasoning and other processing to achieve better, more integrated architectures? How does context come into play when comparing these diagrams, both within an architecture, and especially, across architectures? What techniques and methods exist to capture and relate the semantics contained in these UML diagrams? Is an ontology (or some other formal knowledge representation) a prerequisite to establish context?

Background

The relationship between UML and ontologies has been of interest to the knowledge engineering community for some time. Many of these efforts have focused on using UML as a means for developing ontologies, with less emphasis on extracting semantic data from existing UML diagrams [3-6]. There are also efforts aimed at transformation [7], and efforts to develop tools to support closer interaction between UML and ontologies [8,9]. One project has developed code to convert Rational Rose petal files into RDF [10].

On the DoDAF side, TopQuadrant is developing a set of ontologies specifically for the DoDAF [11], although they are not (yet?) available for public release. These appear to be built using Protégé; no mention is made of UML.

What may be the most comprehensive effort is being led by the Object Management Group (OMG), which is sponsoring a Model Driven Architecture (MDA) based initiative to represent the semantics of ontologies (including, but not limited to OWL) in a MOF2 compliant metamodel, called the Ontology Definition Metamodel (ODM). Also included in this initiative is a requirement for a UML Profile that extends the UML2 metamodel to support ontology definition. The ODM will therefore act as a bridge between UML models and ontologies. This effort holds forth the prospect that automated conversion between the two may be available in the near future, at least as far as the fundamental differences between the two will allow. The OMG has issued an RFP for the ODM, which industry has responded to [12].

Discussion

If extracting the semantic data from UML diagrams were sufficient to provide all of the information needed for architecture integration, then a purely mechanical conversion process would be all that was needed. However, context plays an important role in determining which semantic interpretations are valid and under what circumstances. For example, in the simple activity diagram below, operational activity 1.1 requires information item 1 in order to be completed, and produces information item 2.

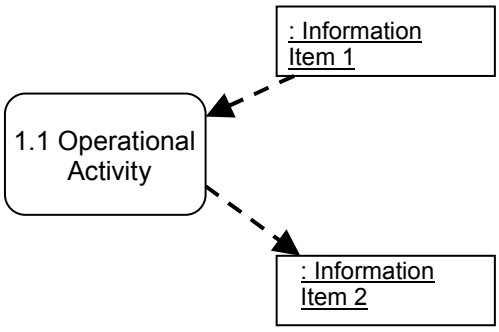


Figure 1 – UML Activity Diagram Fragment

The semantics of this diagram can be extracted, but does it follow that the behavior applies in all situations where operational activity 1.1 occurs? If not, where and how is it defined? Also, at what level of detail does an architect need to represent operational activity 1.1 (and information items 1 and 2) in order to make such analysis feasible? It is

very possible that the same activity may occur in another architecture under a different name.

To support the DoDAF objective of relating architectures across organizational and mission area boundaries it is clear that:

- comparison and integration of architectures must be done in context and with some level of formalism
- ontologies (or some other KR representation) are needed to express context and formalism
- architecture integration will require the merger (and reconciliation) of each architecture's formal representation

Obstacles preventing the integration of architectures include the lack of formalism and context in UML diagrams, effects of different architecting style (subtle variations of style among architects and the existence of legacy diagrams), and organizational barriers that prevent architects from working together. Recently, the DoD has seen the formation of a number of Communities of Interest (COIs) to encourage the development of common vocabularies and shared services. Perhaps these will be convenient vehicles for defining the semantics necessary for architecture integration.

The DoD has made a substantial investment in architecture efforts. In order to achieve its objective it is necessary for these efforts to work together synergistically. This represents a wide open area of research and application for semantic web technologies.

References

1. See http://www.defenselink.mil/nii/doc/ for resources concerning the DoDAF
2. from http://www.defenselink.mil/nii/doc/DoDAF_v1_Memo.pdf
3. K. Baclawski, M. Kokar, et al, "Extending UML to Support Ontology Engineering for the Semantic Web", 2001 available at http://ubot.lockheedmartin.com/ubot/papers/publication/UMLOntology.pdf
4. P. Kogut, S. Cranefield, et al, "UML for Ontology Development", 2001, available at http://ubot.lockheedmartin.com/ubot/papers/publication/KER4.doc
5. S. Cranefield, "UML and the Semantic Web", 2001, available at http://www.semanticweb.org/SWWS/program/full/paper1.pdf
6. D. Gasevic, D. Djuric, V. Devedzic, "Converting UML to OWL Ontologies", 2004, available at http://www.www2004.org/proceedings/docs/2p488.pdf
7. K. Falkovych, M. Sabou, H. Stuckenschmidt, "UML for the Semantic Web: Transformation-Based Approaches", 2002, available at http://homepages.cwi.nl/~media/publications/UML_for_SW.pdf
8. Components for Ontology Driven Information Push (CODIP) project – creators of the DAML-UML Enhanced Tool (DUET), see http://projects.semwebcentral.org/docman/view.php/50/8/DUET_Guides_V0.5.2.doc
9. Protégé UML Backend – available at http://protege.stanford.edu/plugins/uml/
10. Xpetal – part of the Xmodel XML/RDF modeling Tools, available from SourceForge at http://sourceforge.net/projects/xmodel/ [the author is actually pursuing a similar approach using python to parse petal files for import into an ontology]
11. Beta screenshots available at http://www.topquadrant.com/tq_ea_solutions.htm
12. The response to the RFP is available at http://codip.grci.com/odm/draft/submission_text/ODMPrelimSubAug04R1.pdf