Syntactic-Level Ontology Integration Rules for E-commerce

Borys Omelayenko

Vrije Universiteit, Division of Mathematics and Computer Science
De Boelelaan 1081a, 1081 HV, Amsterdam, The Netherlands
www.cs.vu.nl/~borys
borys@cs.vu.nl

Abstract

Electronic marketplaces, or e-commerce portals, bring together many online suppliers and buyers. Each individual participant can potentially use his own format to represent the products in his product catalog, and mapping between them becomes non-trivial. Complicated products require knowledge-intensive descriptions, or ontologies, and catalog integration shifts to integration of product ontologies. The industrial experience analyzed in the paper shows that in some cases the marketplaces require syntactic-level integration of product ontologies, in which the integration rules are created and updated (semi)automatically. Ontology integration tools that satisfy these requirements have not yet been developed, and accordingly we sketch a framework for automated ontology integration that is able to fulfill these requirements.

1. Introduction

Electronic marketplaces, or e-commerce portals, bring together many online suppliers and buyers. Each individual participant can potentially use his own format to represent the products in his product catalog. Complicated products require knowledge-intensive descriptions, or ontologies. Thus, catalog integration requires integration of product ontologies. If a marketplace mediates between *n* suppliers and *m* buyers, then it must be able to map each of the *n* suppliers' catalogs into *m* buyers' formats performing *nxm* mappings. The numbers *n* and *m* can be high enough to make the problem of creation and maintenance of these catalog integration rules non-trivial.

Management of product ontologies and product catalogs occur as a subtask of knowledge management done by the companies. In consequence, it becomes an important part of the ontology-based knowledge management tools, which are now under development within the OntoKnowledge project (www.ontoknowledge.org).

The three types of e-commerce mediation: Business-to-Business (B2B), Business-to-Customer (B2C), and Customer-to-Customer (C2C) differ in terms of number of catalogs, speed requirements, and integration quality. These differences produce different requirements for product integration.

Inference mechanisms developed by the knowledge engineering community provide a standard way to integrate the ontologies. But in B2C and C2C areas the number of catalogs can be in the same order as the number of customers. This makes inference relatively expensive and no longer efficient.

In this paper we consider the problem of ontology integration applied to the task of product integration. In section 2 we survey the requirements for ontology integration that come from the industries; and in section 3 we survey the existing tools for ontology integration. The paper continues with a sketch of the automated ontology integration tool in Section 4, before arriving to final conclusions.

2. The Requirements for Ontology Integration

The three types of electronic commerce, B2B and B2C extensively discussed in (Fensel 2001), and C2C provide different requirements for the integration of product ontologies as surveyed below.

B2B

In the B2B area the numbers of formats *n* and *m* are relatively small, because both correspond to the number of companies. Once created and verified, a rule will be applied many times to a large number of product descriptions. These rules can be constructed manually and must be carefully verified. This gives us the first requirement:

(1) The rules must be understandable by a domain specialist, who may not be a technical expert.

B2B suppliers provide their catalogs in a syntactically unified way, where XML becomes a de-facto standard, and several standards for product descriptions have already been proposed (Li 2000). This brings us to the following requirement:

(2) The rules must be able to translate XML representations of product catalogs.

B2B participants tend to sell more and more complicated products. As a result, the corresponding product ontologies become very complicated. Integration

In: Proceedings of The 14th International FLAIRS Conference (FLAIRS-2001), Key West, FL, May 21-23, 2001,
(c) AAAI Press

is not only required at the instance level, but at the schema level as well:
(3) The rules must be able to deal with ontology schemas: classes, attributes, inheritance, etc.

Electronic product catalogs can be used not only for trade mediation, but also to improve the supply chain used by a company. The paper (Baron, Shaw, and Bailey 2000) discusses implementation of e-catalogs to support the information flows in the supply chain used by the company, its suppliers and customers (B2B procurement) and arrives at the following requirements for product ontology integration:
(4) It must perform integration with the representations used by the legacy systems, which usually have a flat structure and based on the database technology.
(5) The integration must be compatible with security services used in the company.

B2C

The B2C area assumes participation of a large number of individual customers, which can easily reach the order of millions. The product catalogs are customer-oriented and tend to use textual and graphical representation of the products rather than formalized XML descriptions. Accordingly:
(6) Catalog integration rules must be able to interact with wrappers that will translate ill-formalized descriptions into XML.

Many of the requirements are necessitated by the presence of many individual customers and are similar in both B2C and C2C areas.

B2C and C2C

In C2C mediation, suppliers and buyers are represented by individuals, who sell or buy goods. This means that the number of participants can be very high (it can even reach millions) while each participant can use his own format to represent his product. They tend not to use XML, but graphical or textual representations. This imposes special requirements for catalog integration, linked with web site development and customer assistance.

Web-site development for e-business already feels the need to customize product descriptions to the views used by the customers (Santesmases et al. 2000) and requires the integration process to be:
(7) Automatic because each customer requires special integration, and
(8) Easily adaptable to changes in data formats.

The rapidly growing number of suppliers available on the Web has inspired the development of intelligent sales assistants for the Web, able to consult the customers and guide them through the maze of product catalogs and online shops (Traphoner 2000). These service bring with them the following requirements:
(9) Product ontologies must be standardized on the syntactical level with XML.
(10) Product ontologies must be 'derivable', e.g. composed from its technical and market descriptions.
(11) Specific product ontologies must be integrated with general domain ontologies.

For its catalog administration tool, Cohera Corp. (Cohera 2000) works with two different types of catalogs. *Static* catalogs contain slowly changing information that is uploaded and updated periodically and, possibly by several vendors. Meanwhile *dynamic* catalogs that can change on the fly and typically reside at each seller's catalog system. This requirement is quite unusual for manual of semiautomatic knowledge management.

The requirements (Cohera 2000) for the integration system are as follows:
(12) It must be able to integrate the ontologies from multiple and remote sources, where meta-models and inference can be difficult.
(13) Static ontologies have to be integrated together with dynamic ontologies.
(14) It must be able to deal with different expressiveness in the ontology representation languages.
(15) Integrated ontologies will require creation of new categories over the source ontologies, which can lead to generation of the descriptions for 'virtual' products.

This requirement is also unusual for knowledge management: creation of a new super-class must be well-justified because it has to correspond a product (even if it is virtual).
(16) The rules and patterns used to integrate the ontologies must be able to evolve and easily adapt to changes in usage patterns and business relationships.

Thus, the B2C and C2C areas require simple and highly automated techniques at the class-attribute name level. These requirements differ slightly from expressed by (Ng, Yan and Lim 2000), who argue for the development of a simple, scalable and fully automated schema integration technique for B2B and B2C e-commerce**.**

3. Existent Ontology Integration Tools

In this section we survey the tools and algorithms available to the industry in the field of ontology integration. In principle we can perform two types of concept integration: *concept-level integration*, and *syntactical-level integration*. Concept-level integration requires inference over the domain ontology to make a decision about integration of a particular pair of classes. In addition, it also requires an integration ontology that captures the knowledge about the methods able to integrate the classes. Both ontologies are not available explicitly and so the tools require a human expert to make these decisions.

Syntactical integration defines the rules in terms of class and attribute names used in the ontologies to be integrated. Such integration rules are conceptually blind but are easy to develop and implement. This level is widely used in the database community for database schema integration and has proved to be sound (Batini, Lenzerini, and Navathe 1986).

Model-based semantic integration (Bowers and Delcambre 2000) works on a level of semantic models that provides more rationale and flexibility for rules. However, in the B2C and C2C areas customer-oriented representations often come without their underlying models. Furthermore, these algorithms presently concentrate on manual ontology integration and provide no method capable of integrating the ontologies automatically.

Two ontology integration tools have been developed in the knowledge engineering community: Chimaera (McGuinness et al. 2000) and PROMPT (Noy and Musen 2000). Both tools support the merging of ontological terms (class and attribute names) from varied sources. During the class merging process they present the user with pairs of classes whose names are similar enough and might represent either the same class from different input ontologies, or might require taxonomic edition to make one a subclass of the other. A human expert then decides what integration operation to apply to the pair of classes, and the system guides him to the next pair. PROMPT provides more automation in ontology merging. For each ontology merging operation PROMPT suggests the user to perform a sequence of actions on copying the classes and their attributes, creating necessary subclasses and putting them in the right places in the hierarchy. The action sequences are hard-encoded into the system, but experiments showed that they perform very well.

In both approaches an expert still has to decide which ontology integration operation to perform for each pair of classes. This does not correspond to the needs of the industries that require automatic ontology integration.

Recently the database community provided the algorithm for (semi)automatic database schema integration (Palopoli et al., 2000). This paper presents two techniques able to integrate and abstract database schemes. These techniques assume the existence of a collection of relations between the schema attributes, like synonymy, homonymy, hyponymy, a dictionary of overlappings, and a type conflict dictionary. Normally this set of dictionaries does not exist and its construction requires a large investment of time and human effort. Natural language ontologies, like WordNet (Fellbaum 1998) provide a valuable information source for creation of some of these dictionaries and it is essential that the ontology integration algorithm make use of them.

4. Automated Ontology Integration Tool: the First Sketch

The product concepts are represented with classes that correspond to the products or classes of products, and with class attributes that correspond to the product properties.

The input of the algorithm is two sets of classes, each of which corresponds to one ontology to be integrated; the set of class names with the `SubclassOf` relation between them. Each class has a set of attributes associated with the class, where each attribute is represented by its name.

The algorithm has to integrate two ontologies O_1 and O_2, where the first ontology contains the set $O_1 = \{c_1,...,c_{n1}\}$ of classes, and the second has the set $O_2 = \{c_1,...,c_{n2}\}$ of classes. Each class c_i has the associated set of its attributes $A_i = \{a_1,...,a_{mi}\}$. Within this section we will use c_i and a_j to denote the name of a class or an attribute, correspondingly, and C_i to denote the class, or the concept.

Naturally, humans perform incremental ontology integration, comparing the classes one by one, opposite to the batch model, where the expert is supposed to analyze all the classes at once. For automatic integration we have a choice either to perform incremental or batch integration. We will consider incremental integration in this paper, but the batch integration model also has to be considered. From the root of O_1 the algorithm runs an exhaustive (breadth-first or depth-first) search through O_1. On each iteration it compares the current class c_1 from O_1 with all classes from ontology O_2. Consequently, for each pair of classes $c_1 \in O_1$ and $c_2 \in O_2$ the algorithm performs a comparison step described in the next three subsections: it compares the class names, then compares the sets of attributes associated with the two classes, and then compares individual attributes of the classes.

4.1. Integration operations

The integration rules will lead to one of the integration operations to be performed. Until now no unified set of operations has been proposed. The operations available in the literature (Sofia Pinto et al. 1999) are quite general and cannot be used for automated integration. The closest approach to ours, PROMPT, uses the following five merging operations: merge classes, merge slots, merge bindings between a slot and a class, perform a deep copy of a class (with its subclasses and referring classes), perform a shallow copy of a class (only the class itself).

In our framework we can use the following operations:
- Merge classes with the union of their attributes;
- Create a superclass over the pair of classes with the attributes that both classes have in common;
- Rename the classes to fix class name collision;
- Mark the classes as Disjoint;
- Add a subclass-of relation between a pair of classes;
- Remove a subclass-of relation between a pair of classes.

4.2. The Framework for Automated Ontology Integration

During the comparison step we generate the hypotheses about the integration operation required by the pair of classes. Each syntactical feature of the classes, i.e. similarity of their names, can produce a hypothesis about which operation to apply. It is quite possible that several features will produce several different hypotheses. For example, suppose that the class Printer from O_1 with the attributes Technology, Resolution, Interface must be compared with the class The_Printer from O_2 that describes the device with the attribute PrintingTechnology.

Table 1. The hypotheses derived from comparison of the class names

	Condition	Description	Hypothesis
(1)	$c_1 = c_2$	The names are literally equal, then	Merge C_1, C_2
(2)	$c_1 \neq c_2$	**else**	
(3)	$c_1 \subset c_2$	c_1 is a substring of c_2	Make C_2 a subclass of C_1
(4)		**else use domain ontology:**	
(5)	$C_1 \approx C_2$	Synonymous classes	Merge C_1, C_2
(6)	$C_1 > C_2$	C_1 refers to a more general concept than C_2	Make C_2 a subclass of C_1
(7)	$C_1 \parallel C_2$	C_1 and C_2 are disjoint	Mark C_1 and C_2 as disjoint
(8)	**else**		No action

Table 2. The hypotheses derived from comparison of the sets of attributes

	Condition	Description	Hypothesis
(1)	$\lvert A_1 \rvert > \lvert A_2 \rvert$	C_1 is more detailed than C_2.	Restriction: C_2 can not be a subclass of C_1
(2)	$A_1 \subset A_2$	All attributes of C_1 are included in C_2	Make C_2 a subclass of C_1

Table 3. The hypotheses derived from comparison of the individual attributes

	Condition	Description	Hypothesis
(1)	$a_1 = a_2$	The names of the attributes are literally equal	Duplicate class comparison hypothesis
(2)	$a_1 \neq a_2$	**else**	
(3)		**use domain ontology:**	
(4)	$a_1 \approx a_2$	The names are synonymous	Duplicate class comparison hypothesis
(5)	$a_1 \subset a_2$	a_1 is a substring of the name of a_2	No action
(6)	**else**	Just two different attributes	No action

The comparison of the class names will produce the hypothesis that the classes should be merged because their names are semantically equivalent, but the comparison of their attributes might indicate that The_Printer must be defined as a superclass of Printer.

4.3. The Algorithm

The algorithm assumes that some kind of domain ontology of terms exists. It must represent the dictionary used in the domain to recognize the 'synonyms' relation (e.g. Monitor and Display) and the 'is-a' relation between the language concepts (e.g. General_Printer is more specific than Laser_Printer).

Name preprocessing. Before comparing the ontologies some lexical naming confusions must be eliminated at a name preprocessing stage. At this stage for each class name or attribute name the algorithm will:

- Remove the articles, e.g. The_Printer which is equal to Printer;
- Remove encoding prefixes included by a programmer, e.g. strPrinterName which indicates that the attribute PrinterName has string type.
- Separate several words merged to create an identifier, e.g. Laser_Printer, LaserPrinter, and even laserprinter.

An ad-hoc solution that exploits language ontologies like WordNet (Fellbaum 1998) or its domain-specific modifications, can be easily developed for this stage and will perform very effective.

Comparing Class Names. On this stage we compare the names c_1 and c_2, as listed in Table 1. The symbols class names c_1 and c_2 are literally compared with the operations $=$, $\neq$, and substring inclusion $\subset$ as shown in lines (1)-(3). Class concepts C_i are compared in lines (5)-(7) with the operations $>$ that refers to a "more general than" relation, $\approx$ that refers to a "synonymous" relation, and $\parallel$ that stands for a "disjoint" relation, i.e. laser_printer, ink_printer, matrix_printer.

We expect that case (8) will occur most often: the algorithm compares each possible pair of classes from the pair of ontologies, and most of the pairs will not require any operations.

Comparing Attribute Sets. After comparing class names we can compare attribute names. Generally, attributes provide more information than the class name. First, the algorithm takes a look at the attribute sets, and then continues by comparing individual attributes. Possible situations with the attribute sets A_1 of the class c_1 and A_2 of the class c_2 together with the generated hypotheses are listed in Table 2.Operator $\lvert A \rvert$ used in line (1) returns the number of all attributes of the class, including the attributes that were inherited from the superclasses.

Comparing Attributes. Comparing individual attributes gives us additional hypotheses about the integration operation to use. At this stage the algorithm passes each possible pair of attributes $a_1 \in A_1$ and $a_2 \in A_2$ that belong to the classes c_1 and c_2 correspondingly. Their

comparison gives several additional hypotheses as listed in Table 3.

Recently developed ontology representation languages (i.e. RDF (Lassila and Swick, 1999)) include subslot-of relation between the attributes and allow to build hierarchies of attributes similar to class hierarchies. Full integration of such ontologies will also require integration of attribute hierarchies, and extending of Table 3 with new cases.

Making the Decision. For each pair of classes the algorithm will generate the set of possible integration operations to be performed over the pair of classes (the hypotheses). The algorithm generates all possible pairs of classes for comparison, and we expect that most of the pairs will require no integration. To find the class $c_2 \in C_2$ that has to be merged with the class $c_1 \in C_1$ the algorithm will pass all n_2 classes from C_2 and $n_2 \cdot m$ attributes, where m is the average number of attributes of a class from ontology O_2, and has to perform only one merging operation. For a realistic case of $n_2 = 50$ and $m = 7$ this will give 350 tests, that can lead to up to 350 hypotheses. The decision-making step must select only one integration operation out of these hypotheses, and lots of 'noisy' hypotheses must be ignored.

The opinion aggregation algorithm has a user-adjustable threshold: the number of generated hypotheses must exceed some predefined threshold for the algorithm to perform an integration operation. The hypotheses are aggregated with simple voting, a method with a long history that approved its robustness in many application areas. Thus, if the comparison of two classes from a pair produces three hypotheses 'merge' and five hypotheses 'make c_1 a subclass of c_2' then the algorithm will perform the second operation.

5. Conclusions

The paper shows that, in the case of B2C and C2C, the e-commerce requires syntactic-level integration of product ontologies, where the integration rules are created and updated (semi)automatically. There are no ontology integration tools that satisfy these requirements, and the closest approaches force the user to make a decision how to integrate a certain pair of classes. The proposed integration algorithm attempts to automate this decision-making process.

Acknowledgement. The author would like to thank to Dieter Fensel for helpful discussions and comments on this paper.

References

Baron, J.; Shaw, M.; Bailey, A. 2000. Web-based E-catalog Systems in B2B Procurement, *Communications of the ACM* 43(5):93-100.

Batini, C.; Lenzerini, M.; Navathe, S. 1986. A comparative analysis of methodologies for database schema integration, *ACM Computing Surveys* 18(4):323-364.

Bowers, S.; Delcambre, L.; 2000. *Representing and Transforming Model-Based Information*. In: Proceedings of the Workshop on the Semantic Web at ECDL-00, Lisbon, Portugal, September 21.

Cohera Corp. 2000. *E-Catalog Integration for E-Markets*. www.cohera.com.

Fellbaum, C. 1998. *WordNet: An Electronic Lexical Database*. The MIT Press.

Fensel, D. 2001. *Ontologies: Silver Bullet for Knowledge Management and Electronic Commerce*. Springer-Verlag, Berlin, 2001.

Lassila, O.; Swick, R. 1999. Resource Description Framework (RDF) Model and Syntax Specification, *W3C Recommendation*, Feb. 1999; available online at http://www.w3.org/TR/REC-rdf-syntax/.

Li, H. 2000. XML and Industrial Standards for Electronic Commerce, *Knowledge and Information Systems* 2(4):487-497.

McGuinness, D.; Fikes, R.; Rice, J.; 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), Breckenridge, Colorado, 12-15 April.

Ng, W.; Yan, G.; Lim, E. 2000, 'Heterogeneous Product Description in Electronic Commerce', *SIGecom Exchanges, Newsletter of the ACM Special Interest Group on E-commerce* 1.

Noy, N.; Musen, M. 2000. *PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment*. In: Proceedings of the AAAI-00 Conference, Austin, TX.

Palopoli, L.; Pontieri, L.; Terracina, G.; Ursino, D. 2000. Intensional and Extensional Integration and Abstraction of Heterogeneous Databases, *Data & Knowledge Engineering* 35:201-237.

Santesmases, J.; Lopez, D.; Fernandez, Y. 2000. *SPGC: A Personalized Server and Content Manager*. In: E-business: Issues, Applications and Developments. Proceedings of eBusiness and eWork 2000 Conference and Exhibition, Madrid, Spain, 18-20 October.

Sofia Pinto, H.; Gomez-Perez, A.; Martins, J. 1999. *Some Issues on Ontology Integration*. In: Proceedings of the IJCAI-99 workshop on Ontologies and Problem-Solving Methods (KRR5), Stockholm, Sweeden, 2 August.

Traphoner, R. 2000. *WEBSELL: Intelligent Sales Assistants for the World Wide Web*. In: E-business: Issues, Applications and Developments. Proceedings of eBusiness and eWork 2000 Conference and Exhibition, Madrid, Spain, 18-20 October.