Metalevel for an efficient query answering

Bermúdez J., Illarramendi A., Blanco J.M., Goñi A.

Facultad de Informática, Universidad del País Vasco. Apdo. 649, 20.080 San Sebastián. SPAIN
e-mail: jipbeanj@si.ehu.es

1 Introduction

The integration of heterogeneous and autonomous information sources is a requirement for the new type of cooperative information systems. Heading to this goal we have built a system that allows the integration of heterogeneous and autonomous relational databases using a terminological system ([?], [?], [?]).

The databases integration process begins with a translation step. A semantically rich terminological representation of each relational schema to be integrated is created. It is followed by a proper integration step.The integrated schema is generated by first, defining correspondences among data elements (concepts and roles) of the terminologies that must be integrated and then, applying some integration rules. Moreover, after translation and integration a mapping information that links data elements of the integrated schema with the underlying databases is also generated. Once an integrated schema has been obtained, it provides the users with an integrated and global view of the data stored in pre-existing databases, that can be used to formulate queries over. The goal of the query processing step is to find the answers to these queries in an efficient way. Four different tasks are involved in the query processing step: (1) parsing of the query, (2) semantic optimization, that is, transforming the query into another one with the same answer, but that can be processed more efficiently, and detection of inconsistencies, (3) identification of the cached parts of the query and if possible looking for the answer in the cache memory, and in other cases (4) generation of an optimal plan to obtain the non-cached parts of the query from the underlying databases. This lat task involves the translation of the query into relational queries over different databases. In this paper we present an alternative approach to the fourth task, based on the classification culture and treating to take advantage from well known techniques in other areas of systems programming.

We propose a metalevel definition mechanism to classify metaconcepts that will represent relevant features of concepts and roles from the plan generation point of view. Each metaconcept will be associated with an ad hoc generic plan. A proper instantiation of that plan will efficiently generate the answer to a query belonging to that metaconcept. The relevant features of concepts and roles that must be described are (1) the syntactic structure of its definition, and (2) the semantic interrelations satisfied by their components. The aim of this paper is to present a description language developed for those metaconcepts, and a subsumption notion associated to it.

Other works that deal with the integration task ([?], [?], [?], [?], [?]) do not follow this approach. Moreover, as far as we know, the only terminological system that supports a metalevel notion is Omega([?], [?], [?]). However, it exceeds our needs, and its powerful expressiveness makes subsumption undecidable, which is inappropriate for our goal.

In the rest of this paper we present our proposal of a description language for metaconcepts. We start defining the basic units and a hierarchical organization among them.Later,we

<basic pattern> ::= ANYCONCEPT
| ANYROLE
| NAMED-CONCEPT
| NAMED-ROLE
| **prim** <pattern name>
| π(var::<basic pattern>)*.<term pattern>
<term pattern> ::= <operator> (<arg>)*
<operator> ::= <symbol>
<arg> ::= <var> | <value>

Figure 1: Basic Patterns

introduce composite expressions of metaconcepts and the notion of subsumption.Finally, we discuss some application examples.

2 The Proposal

As said in the introduction, our work is guided by our interest in introducing a classification approach in the task of plan generation for the query processing step. The next subsections present the main features of the metalevel description language according to the relevant features of terms[1] stated above:(1) syntactic structure and (2) semantic interrelations.

2.1 Basic patterns

Concerning the syntactic structure, patterns specify syntactic categories -metaconcepts- to which the terms can belong to. The basic patterns have been extracted from the collections of term constructors of the terminological language considered. For instance, the basic pattern πr::ANYROLE,c::ANYCONCEPT.**all**(r,c) defines the syntactic category of the *all* concepts. π binds variables to form parameterized patterns in which the free variables of the term-pattern are just those bounded by π. We also consider as basic some primitive patterns that cannot be specified by syntactic features alone. For example, **prim** FUNCTIONAL-ROLE defines the syntactic category of the roles that are restricted to be functions. Moreover, there is a similitude between primitive patterns and primitive concepts. In both cases, its membership is usually asserted. For instance, **prim** CACHED defines the syntactic category of concepts and roles that have precalculated extensions. That is, those terms for which we don't need to elaborate a plan to discover their individuals because their extensions are cached. Notice that this information is desirable in a context of efficient retrieval. Membership to **prim** CACHED metaconcept must be asserted because this information is non-definitional.

Figure 1 defines the basic pattern category. The constants ANYCONCEPT and ANY-ROLE represent the universal metaconcepts of concept and role terms respectively.The NAMED-CONCEPT constant is a subclass of ANYCONCEPT which include the concepts that are denoted by a name, in contrast with those denoted by expressions. The same applies to NAMED-ROLE and ANYROLE. Thus, all concepts of the integrated terminology are in NAMED-CONCEPT because there is a name associated to their definitions. Bounds of variables act as types. For basic patterns only constant and primitive patterns are allowed as bounds.

Every operator of a parameterized pattern has an attribute associated: the *type* attribute which tell us the primitive or constant pattern of a term that matches the parameterized

[1] Hereafter we will use the word terms to refer to concepts and roles.

pattern definition. For example *type* (**all**)=ANYCONCEPT. The basic patterns are arranged in a partial order hierarchy which is defined as the least order relation $\preceq$ respecting the following:

- ANYCONCEPT and ANYROLE are maximal with respect to $\preceq$.
- NAMED-CONCEPT $\preceq$ ANYCONCEPT and NAMED-ROLE $\preceq$ ANYROLE.
- For every parameterized pattern P with operator op: P $\preceq$ *type*(op).
- When defining a primitive pattern P, at most one basic pattern Q can be related as P $\preceq$ Q.

For example:

- πr::ANYROLE,c::ANYCONCEPT.**all**(r,c) $\preceq$ ANYCONCEPT
- **prim** FUNCTIONAL-ROLE $\preceq$ ANYROLE
- **prim** CACHED (is maximal with respect to $\preceq$)

Furthermore, we have defined a binary relation *is inconsistent with* between basic patterns. P *is inconsistent with* Q if it is impossible for them to share an individual. This relation is antireflexive, symmetric and satisfies a sort of transitivity: P $\preceq$ Q and Q *is inconsistent with* R implies that P *is inconsistent with* R. By definition, ANYCONCEPT *is inconsistent with* ANYROLE, and every parameterized pattern is *is inconsistent with* each other. We say that two basic patterns P, Q are consistent if not P *is inconsistent with* Q.

2.2 Complex patterns

Basic patterns are the units to build up complex patterns. Parameterized patterns properly connected can be represented as trees: the operators are nodes and variables are labels for the edges connecting their corresponding arguments (there is an edge for each variable appearing in the introduction of the operator). The leaves of these trees are the constant or primitive patterns that serve as types for the variables, or the values that instantiate the variables.

Patterns can be seen as formal language specifications. Figure 2 shows the operations selected for these specifications. The $\cup$ and $\cap$ operators describe the union and intersection, respectively, of the corresponding language specifications. π-expressions are generalized parameterized patterns where the free variables of the body are just those in the declaration part[2]. Those expressions are fundamental for composing specifications whose meaning is the composition of the meanings of the parts. Restriction on a π-expression either substitutes variables for instance values or restricts the type of a variable to a more specific pattern (a subsumee of the substituted pattern). Constraint addition specifies those terms of the pattern expression part that satisfy the constraint part. The allowed constraints are predicates that correspond to the functionalities provided by the underlying terminological system. We also admit definitions, including recursive ones, that associate a name to a pattern. Figure 3 shows some examples.

Considering the previous model, subsumption between patterns becomes inclusion between languages. Pattern Q *subsumes* pattern P (P $\sqsubseteq$ Q) if the language specified by P is a subset of that specified by Q. Basic patterns form the skeleton hierarchy upon which we build the subsumption hierarchy. For basic patterns P, Q: P $\preceq$ Q implies P $\sqsubseteq$ Q. The $\cup$ and $\cap$ operators specify, respectively, the least upper bound and greatest lower bound of their operands with respect subsumption. ANYCONCEPT $\cup$ ANYROLE is the maximum of this hierarchy. Composition into π-expressions and restriction are monotonic, and constraint

[2]Technical problems of variable name clashes are not treated here.

```
<pattern>          ::= <pattern name>
                     | <basic pattern>
                     | <pattern> ∪ <pattern>
                     | <pattern> ∩ <pattern>
                     | <π-expression>
                     | <π-expression>(<substitution>)
                     | <pattern> + <constraint>
< π-expression>    ::= π(var::<pattern>) *. <pattern>
<substitution>     ::= <var> ← <arg>
                     | <var>::<pattern>
<constraint>       ::= terminological system services
```

Figure 2: Complex Patterns

```
ALL= π r::ANYROLE, c::ANYCONCEPT. all(r,c)
ATLEAST= π r::ANYROLE, n::NAT. atleast(n,r)
AND= π c1::ANYCONCEPT, c2::ANYCONCEPT. and(c1,c2)
ALL-ATLEAST= ALL(c::ATLEAST)
ONE-ROLE-QUERY= π s::ANYROLE.
                    ALL(r ← s, c::ONE-ROLE-QUERY(s ← s))
                    ∪ ATLEAST(r ← s)
                    ∪ AND(c1:: ONE-ROLE-QUERY(s ← s),
                          c2:: ONE-ROLE-QUERY(s ← s))
DISJ-RANGE= ALL + [range[3]r disj c]
```

Figure 3: Pattern Examples

addition specifies more specific patterns. The binary relation *is inconsistent with* can be extended straightforwardly to patterns adding the following rule: if P $\sqsubseteq$ Q and Q*is inconsistent with* R then P*is inconsistent with* R. The intersection of two inconsistent patterns specifies the empty language of terms.

3 Applications

The person responsible of the integration will be able to define patterns (metaconcepts), that will be classified by the system, and associate with it a plan if desired. Inheritance provides for alternative plans associated with its subsumers. Queries over the integrated terminologies will be recognized as members of patterns. The preferred plan for the answer is the first associated plan encountered, moving upwards the subsumption hierarchy.(Alternative plans will be available in case of multiple inheritance).

This framework allows to tailor the plans, improving the possibilities of semantic transformations or case analysis of queries.

Let us see the plans as SQL queries. Let X-table be the relational table that supports the concept or role named X. Let **key** be the key attribute for the tables supporting concepts and **fst** and **snd** the attributes supporting the tuples of roles.

Example 1: To the metaconcept ALL(r::ANYROLE, c::ALL(r←s,c←d) we associate a plan that avoids nested queries (that would result from the compositional naive translation of the semantics of its concept members).

SELECT ANYTHING-table.**key** FROM ANYTHING-table

[3] Any role has two attributes: *dom* and *range*.

MINUS
SELECT r-table.**fst** FROM r-table
WHERE r-table.**snd**=s-table.**fst**
AND s-table.**snd** NOT IN SELECT d-table.**key** FROM d-table

Example 2: During the integration process, a mapping information is created that allows us to establish that the role *age* is a function. Queries such as **atleast**(5,*age*) will be recognized as belonging to the syntactic category π r::FUNCTIONAL-ROLE, n::NAT. **atleast**(n,r) + [n>1] and the plan associated should be the one that informs about an empty set answer query. Notice that we discover cases where the terminological subsumption algorithm does not respond **atleast**(5,*age*)$\sqsubseteq$ NOTHING.

Example 3: Suppose an integrated terminology where A-ENDED $\sqsubseteq$ LETTER-ENDED, the role social security identifier is defined as *soc-sec* :<**domain**(PERSON) **and range**(KEY-ID) and the terminological system can deduce LETTER-ENDED **and** KEY-ID $\sqsubseteq$ NOTHING. Moreover, the mapping information registers that the role *soc-sec* represents a non null key attribute for persons (that is, *soc-sec* is total on its domain).

A user who ignores the characteristics of the social security identifier may ask for PERSON **and all**(*soc-sec*, A-ENDED). Usual subsumption algorithms cannot discover that there is no person satisfying the description. Nevertheless, the query can be recognized as an individual of the metaconcept

AND(c1::ANYCONCEPT,c2::ALL(r::**prim** DOM-TOTAL, c::ANYCONCEPT))
+ [*dom* r $\equiv$ c1, range r **disj** c]

Notice that, on the one side, the extension of any concept satisfying this pattern are members of c1 whose fillers for r fall in c. On the other side, for those concepts, it must be verified:

1. *dom* r $\equiv$ c1 and r is total on its domain, so r has fillers for any member of c1 and
2. range r **disj** c, so fillers for r are never in c.

Therefore we get a contradiction which implies that the extension is the empty set. The plan associated is the same as the one mentioned in the previous example.

References

[ACHK93] Y. Arens, C.Y. Chee, C. Hsu, and C.A. Knoblock. Retrieving and integrating data from multiple information sources. *International Journal of Intelligent and Cooperative Information Systems*, 2(2):127–158, 1993.

[AS84] G. Attardi and M. Simi. Metalanguage and reasoning across viewpoints. In *Proceedings 6th European Conf. on Artificial Intelligence. Pisa.Italy*, 1984.

[AS86] G. Attardi and M. Simi. A description-oriented logic for building knowledge bases. In *Proceedings of the IEEE, Vol. 74, No. 10*, October 1986.

[BIG91] J.M. Blanco, A. Illarramendi, and A. Goñi. A uniform approach to design a federated information base using BACK. In *Proc. of the Terminological Logic Users Workshop. Berlin*, 1991.

[BIG95] J.M. Blanco, A. Illarramendi, and A. Goñi. Building a federated database system: an approach using a knowledge based system. To appear in the Int. Journal on Intelligent and Cooperative Information Systems., 1995.

[BIGB94] J.M. Blanco, A. Illarramendi, A. Goñi, and J. Bermúdez. Advantages of using a terminological system for integrating databases. In *Proc. of the International Workshop on Description Logics. Bonn*, 1994.

[BS92] S. Bergamaschi and C. Sartori. On Taxonomic Reasoning in Conceptual Design. *ACM Transactions on Database Systems*, 17(3):385–421, 1992.

[BST+93] R.J. Brachman, P.G.. Selfridge, L.G. Terveen, B. Altman, A. Borgida, F. Halper, T. Kirk, A. Lazar, D.L. McGuinnes, and L.A. Resnick. Integrated support for data archeology. *International Journal of Intelligent and Cooperative Information Systems*, 2(2):159–185, 1993.

[HAS80] C. Hewitt, G. Attardi, and M. Simi. Knowledge embedding in the description system omega. In *Proceedings of First National Annual Conference on Artificial Intelligence. Stanford University*, August 1980.

[SGN93] A.P. Sheth, S.K. Gala, and S.B. Navathe. On automatic reasoning for schema integration. *International Journal of Intelligent and Cooperative Information Systems*, 2(1):23–50, 1993.

[SPD92] S. Spaccapietra, C. Parent, and Y. Dupont. Model independent assertions for integration of heterogeneous schemas. *The VLDB Journal*, 1(1):81–126, 1992.