Incorporating new Languages to DL systems

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 great expansion of the communication networks have made available to the users of a huge number of heterogeneous and autonomous data repositories. However, these repositories present different structures/organizations, query languages and data semantics, making very difficult for the users to access the data stored on them.

A possible solution to lighten the problem of lack of uniformity when dealing with the available repositories consists on defining new information retrieval techniques with a strategy that focuses on information content and semantics. We propose to represent intensional descriptions of the objects in the repositories as metadata introducing in this way a semantic view over the repositories.

Different kind of systems can be used to represent those semantic views. We consider that systems based on Description Logics (DL systems) are interesting for that purpose due to the following reasons [BIGB94; GBBI96]: they allow definition of semantically richer views, they are also appropriated to offer intensional answers to the users and last, reasoning mechanisms from DL systems are useful to perform query optimization and in particular semantic and caching optimization.

Moreover, when defining semantic views over repositories it is necessary to include also a mapping information that relates terms of the semantic views (classes and roles in our case) with one or more repositories where the actual data are stored. The mapping descriptions play a key role in encapsulating the heterogeneity due to different formats and organization of the data in the various repositories.

Existing DL systems provide languages for describing classes and roles and also for creating instance objects that represent the beliefs of the system. However, for the considered framework, managing multiple information systems, two more languages, one for defining meta terms and another one for describing the mapping information can be very useful. The goal of the first language is in general to provide extensibility and in our case we use it to describe the syntactic structure and the semantic interrelationships among the components of terms. The goal of the second language, that we are working on, is to allow defining mapping descriptions at the same level as classes and roles.

$$
\begin{aligned}
C, D \rightarrow\ & \top \\
& \mid \bot \\
& \mid \mathbf{feature}(f : C) \\
& \mid s : C \\
& \mid s_1 @ s_2 \\
& \mid s @ \mathit{term} \\
& \mid C \sqcap D \\
& \mid C \sqcup D \\
& \mid \mathbf{disj\_prim}(C, \mathit{index}, \mathit{group})
\end{aligned}
$$

Figure 1: Language for a metalevel

2 A Language for a Metalevel

If terms are going to be instances, which of their properties should be abstracted? Syntactic structure of terms and semantic relationships among their components are relevant aspects to be described. Our metaclass descriptions are based on a collection of functions, that we call *features*, which represent characteristics of the instances of metaclasses. Basically, a term will satisfy a metaclass description if the features specified in the description apply to that term. We propose the language in figure 1, that resembles the language of attribute-value logic in [Car92], for describing metaclasses. Where:

- C and D represent metaclasses.
- $\top$ and $\bot$ are the top and the bottom of the hierarchy respectively.
- **feature**$(f : C)$ introduces a new feature f with $[\![C]\!]$ being the range of its values and:

  $$[\![\mathbf{feature}(f : C)]\!] = \{t \in [\![\top]\!] \mid f(t) \mathit{\ is\ defined}\}$$

- s, s_1, s_2 are feature chains (composition of functions), *term* represents an instance at the class level and @ represents roles (binary relations) defined on terms:

  $$@ :< \mathbf{dom}(C)\mathbf{and\ ran}(D)$$

$$
\begin{aligned}
&[\![@]\!] \subseteq [\![C]\!] \times [\![D]\!] \\
&[\![s : C]\!] = \{t \in [\![\top]\!] \mid \\
&\quad s(t) \mathit{\ is\ defined} \wedge s(t) \in [\![C]\!]\} \\
&[\![s_1 @ s_2]\!] = \{t \in [\![\top]\!] \mid
\end{aligned}
$$

CONCEPT_TERM =
  disj_prim($\top$, *concept_term, term*)
ROLE_TERM =
  disj_prim($\top$, *role_term, term*)
ROLE_REST=
  CONCEPT_TERM $\sqcap$
  feature(role:ROLE_TERM)
RANGE_REST =
  ROLE_REST $\sqcap$
  feature(concept:CONCEPT_TERM)
NUMERIC=
  disj_prim($\top$, *numeric, term*)
NUMBER_REST =
  ROLE_REST $\sqcap$
  feature(num: NUMERIC)
ATLEAST =
  disj_prim(NUMBER_REST, *atleast, opr*)
ATMOST =
  disj_prim(NUMBER_REST, *atmost, opr*)
ALL =
  disj_prim(RANGE_REST, *all, opr*)
ALL_REST =
  ALL $\sqcap$ num $\cdot$ concept:INTERVAL

Figure 2: Metaclasses

$s_1(t), s_2(t)$ *are defined* $\wedge$ $(s_1(t), s_2(t)) \in [\![@]\!]\}$
$[\![s@term]\!] = \{t \in [\![\top]\!] \mid$
  $s(t)$ *is defined* $\wedge$ $(s(t), term) \in [\![@]\!]\}$
$[\![C \sqcap D]\!] = C \cap D$
$[\![C \sqcup D]\!] = C \cup D$
$[\![$**disj_prim**$(C, index, group)]\!] \subseteq [\![C]\!]$

- **disj_prim**(C, *index, group*) defines primitive metaclasses and pairwise disjoint with those of the same *group* parameter (the same operator as in *CLASSIC* [BMPS+91].

Notice that the usual class level becomes now an instance level too. With this language we can define metaclasses so that any term provided by DL systems will be an instance of them (see figure 2).

From this point of view, a language for the definition of terms is an assertional language at the class level. For example, the definition of the term:

$$t := \mathbf{all}(r, \mathbf{atmost}(n, s))$$

asserts the following:

$$t :: \text{ALL} \sqcap \text{concept} : \text{ATMOST}$$

and:

$$\begin{aligned} \text{role}(t) &= r \\ \text{num} \cdot \text{concept}(t) &= n \\ \text{role} \cdot \text{concept}(t) &= s \end{aligned}$$

Moreover, new kind of terms can be added naturally, for example CLASP terms [Bor92] defined to represent classes of actions and plans:

STATE=
  disj_prim(CONCEPT_TERM, *state, sort*)
ACTION =
  disj_prim(CONCEPT_TERM $\sqcap$
      feature(*start* : STATE) $\sqcap$
      feature(*end* : STATE) $\sqcap$
      feature(*add* : STATE) $\sqcap$
      feature(*delete* : STATE), *action, opr*)
SCENARIO =
  disj_prim($\top$, *scenario, term*)
SINGLE =
  disj_prim(SCENARIO $\sqcap$
      feature(*act* : ACTION), *single, opr*)
SEQUENCE =
  disj_prim(SCENARIO $\sqcap$
      feature(*fst* : SCENARIO) $\sqcap$
      feature(*snd* : SCENARIO), *seq, opr*)

Furthermore, roles can be attached to metaclasses. For example, the subsumption relationship between terms could be defined as a role attached to $\top$.

$$\sqsubseteq :< \mathbf{dom}(\top)\,\mathbf{and}\,\mathbf{ran}(\top)$$

but other relationships could also be interesting. For example interschema and intraschema assertions in [CL93]. Subsumption on this metalevel depends on the semantics of (@) roles attached to metaclasses.

3 Language for describing a mapping information

As mentioned before, an important aspect when coupling different types of systems (DL systems and heterogeneous repositories in our case) is the definition of the mapping information. However, mapping languages are not described in detail in the literature. We are working on mapping descriptions that subscribe to the idea of viewing a data repository as a set of entities and attributes, independently of the concrete organization of the data in the repository. Attributes play a central role. We assume there exists an equivalence relationship defined on them, formalizing their semantic equivalence. Therefore we can take representative attributes as unique up to semantic equivalence. We advocate for these attributes being the coordinates of a high dimension vectorial space. Every set of attributes defines a subspace. Each mapping description specifies its corresponding subspace; meaning that the entities they support are identified just by these attributes. When the repository is a relational database, subspace descriptions may include functional dependencies among attributes. Mapping descriptions are tuple sets descriptions. They are expressed with a propositional formula that represents a relational algebra expression, in the following way:

- Every base relation is associated with a propositional constant.
- A select operation on the relation associated with P is represented by $P \wedge f$ where f is a propositional constant that represents the selection condition.

- Union, intersection and difference are represented by $P \vee Q, P \wedge Q, P \wedge \neg Q$ respectively.

All these operations give relations within the same subspace, and propositional formulae implication implies relation inclusion. Inverse implication cannot be guaranteed. Cartesian product renders a relation inside the subspace specified with the union of their attributes (i.e. coordinates) and then receives its associated propositional constant. Conjoining a propositional formula to a subspace focuses on the desired attributes.

Next we show metalevel terms whose instances are mapping descriptions:

MAPPING_TERM =
  disj_prim($\top$, *mapping_term*, *term*)
MAP =
  disj_prim(MAPPING_TERM $\sqcap$
   feature(*space* : ATTRIBUTES) $\sqcap$
   feature(*constraint* : PROP_FORMULA),
     map, *opr*)

Notice that the notion of subsumption in this context involves sets of attributes. A mapping description **map**(A, α) subsumes another one **map**(B, β) if:

- $A \subseteq B$.

  The attributes needed to identify the different instances represented by α is a subset of the attributes needed to identify those represented by β. That is, B can be considered as a superkey for A.

- $\beta \rightarrow \alpha$

Moreover, the role *supporter:*

$\lhd$:< **dom**(CONCEPT_TERM)
  and ran(MAPPING_TERM)

relates classes with the repositories where the actual data are stored.

Finally, once we have a language to describe a mapping information it results very interesting to provide a query language that allows formulating queries such as:

CONCEPT_TERM $\sqcap \varepsilon \lhd$ **map**(A, α)

(ε represents the empty chain)

The answer to that query could be useful for one interested in knowing the repercussions of the deletion operation of attributes in A.

References

[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. Germany*, 1994.

[BMPS+91] R.J. Brachman, D.L. McGuiness, P.F. Patel-Schneider, L.A. Resnick, and A. Borgida. Living with CLASSIC: When and how to use a KL-ONE-like language. In *Principles of Semantic Networks (J.F. Sowa ed.)*, pages 401–456. Morgan Kaufmann Pub. Inc., 1991.

[Bor92] A. Borgida. Towards the systematic development of description logic reasoners: CLASP reconstructed. In *Proc. of Principles of Knowledge Representation and Reasoning (KR'92)*, pages 259–269, 1992.

[Car92] Bob Carpenter. *The Logic of Typed Feature Structures.* Cambridge University Press, 1992.

[CL93] T. Catarci and M. Lenzerini. Representing and using interschema knowledge in cooperative information systems. *International Journal of Intelligent and Cooperative Information Systems*, 2(4):375–398, 1993.

[GBBI96] A. Goñi, J. Bermúdez, J.M. Blanco, and A. Illarramendi. Using reasoning of description logics for query processing in multidatabase systems. In *Proc. of the Knowledge Representation meets Databases (KRDB '96) in ECAI '96. Budapest (Hungary)*, 1996.