Knowledge Representation meets Databases — a view of the symbiosis —

Alex Borgida

Dept. of Computer Science
Rutgers University
New Brunswick, NJ 08904, USA
borgida@cs.rutgers.edu

Abstract. This rather informal paper surveys a personal selection of research projects which addressed new problems related to Databases, and whose solution was both inspired by ideas from the field of Knowledge Representation and Reasoning, and at the same time ended up contributing new ideas to that field. The problems include natural language access to databases, Information System (environment) design, permitting exceptions to integrity constraints, configuration databases, and goal-oriented schema design.

1 Introduction

It is possible to view KR&DB research from multiple viewpoints. First, it can be considered as applying database notions to knowledge representation systems. The typical concerns of database research are taken to be persistence and scale, which require special data storage techniques and possibly optimization of queries. Less frequently considered outside the DB field, but just as important, are notions such as concurrency control and recovery in case of failures. Thus for any KR&R system equipped with a Tell/Ask interface (e.g., a rule-based system, description logic reasoner), there have been investigations on how to add (some of) the above "database properties". For example, Chaudri et al. [7] considered the issue of concurrency control for knowledge bases.

In the opposite direction, one can view KR&DB research as applying KR ideas to traditional database problems. For example, normally the standard first step in database schema design is drawing an Entity-Relationship diagram. It turns out that one can check the self-consistency of ER diagrams, or of single entity sets in them, by translating them into a description logic, and then using a standard DL reasoner, as done by the i.Com tool [9].

I will focus instead on the mutually supportive research at the meet of these two areas. Invoking the privilege of an invited speaker, I will concentrate exclusively on work that I have personally witnessed or been involved in[1].

[1] I have had the great fortune to collaborate on much of this work with John Mylopoulos and Ron Brachman, who are not just outstanding scientists but, even more importantly to me, the nicest, most easy-going and supportive people one could hope to meet. It gives me great pleasure to acknowledge their invaluable help.

To give some shape to my presentation I will use a list of topics that have been the focus of considerable research in the KR&R field:

1. semantic networks
2. objects/classes with $instanceOf$ and IsA
3. First Order Logic
4. Description Logics
5. goals

and discuss how attempting to solve new database-related problems led to solutions inspired by these notions, and frequently new results of interest to KR&R itself. Given the nature of the audience, I will also try to point occasionally to interesting connections to topics such as Description Logics and the Semantic Web.

2 Semantic Networks

In 1974/75 John Mylopoulos was flooded by a large group of new students interested in AI research (including James Allen, Phil Cohen, Hector Levesque, John Tsotsos and yours truly). He proposed to work on the problem of **natural language access to databases** - the Torus project. (Other significant AI research in NLP was being driven by the same problem, including William Woods' LUNAR project, and William A. Martin's EQS/OWL projects.) To solve the NLP problem, one needs, of course, a representation of the semantics of sentences, and this being the mid 1970's, the answer was semantic networks: labeled directed graphs, which in our case had an open-ended label set for nodes, but only employ a fixed pre-determined set of possible edge labels.

Fig.1 is based on [14], and illustrates part of a semantic network graph describing the process of writing, sending and receiving recommendation letters.

What makes this interesting from a KR&DB point of view is that in order to answer questions such as "Did we receive any letters for Jimbo?", the semantic representation needs to be connected to the database. Supposing that there is a database table

`RecLetterTable(Name,Source,Address,Text,DateReceived)`

this is accomplished by connecting every column to a node in the graph (e.g., `Name` to `Applicant`, `Source` to `Recommender`, ...). The result is that *the semantic network acts as a semantic data model for the database* — one providing, in fact, considerably richer semantics than standard ER diagrams. What was missing of course, as elsewhere in AI, is precise semantics for semantic networks.

KR&R Resonances

The following quotes from the IJCAI'75 papers on Torus resonate interestingly with later topics in the field of DL:

– *"Due to the properties of the sub/superset hierarchy, there is a unique position in the semantic net for each semantic graph we wish to integrate"*. For

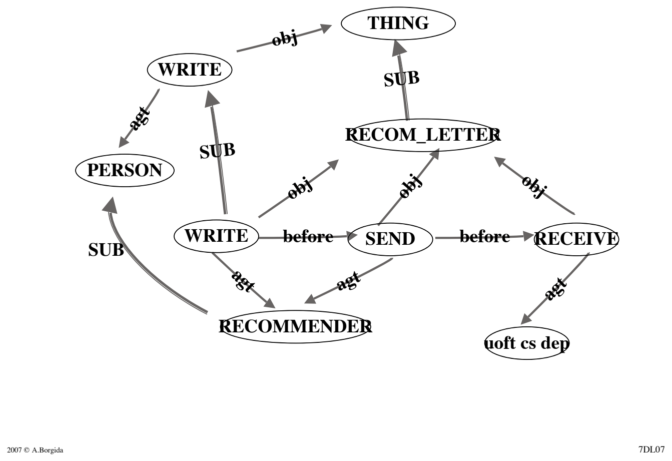

Fig. 1. Part of a Torus Semantic Network

example, as illustrated in Fig.1, "recommender writing a recommendation letter" is a specialization of "writing". Thus Torus' (graph) classification resembles concept classification in DLs, but without a clear idea of what is a definition vs primitive, or what is the semantics of subsumption.

– The term *"definitional axis"* is mentioned in [15], but with no further explanation.
– Torus represents properties, such as `hasColor`, using so-called "characteristics":

```
 PHYS_OBJECT <--ch-- COLOR --val--> COLOR_VALUE
      /                                 /
     /                                 /
   eltOf                            eltOf
    /                                 /
PapaSmurf <--ch-- COLOR --val-->  Blue
```

which resembles the representation suggested in the recent Dolce ontology [10], based on the notion of *qualia*.

– Some concepts, like `ADRESS_VALUE`, have an associated "recognition function" to recognize instantiations, such as `'65 st george street, toronto'`. This prefigured the *test-defined concepts* of the Classic description logic [5] (and of Taxis [16]): these are concepts that have associated procedures for recognizing instances, so that they support instance classification, but are treated as primitive "blackbox" concepts as far as concept classification.

3 Inheritance and Meta-classes Everywhere

In 1977, Mylopoulos and Wong, with the assistance of Phil Bernstein, embarked on the Taxis project [16], whose goal was to develop **a language for designing/implementing Information Systems** at a conceptual, rather than "logical"/"physical" level. Taxis provided a conceptual modeling language not just for objects but also for procedures, exceptions, exception handlers, and eventually workflows. So, for example, after specifying Students, Courses, and how to Enroll students into courses, one could describe Part_Time_Students, Graduate_Courses, and additional details concerning the procedure to enroll for these specialized arguments. My participation in the project concerned the formal semantics of Taxis, especially procedures, workflows ("scripts"), and the overall methodology of programming by specialization.

KR&R Resonances

- IsA hierarchies of procedures were also a new idea in KR&R — see [2].
- Taxis scripts [1] which prefigured workflows, were also organized in inheritance hierarchies. This echoes somewhat the application of DLs to workflow fragment management [11].

While Taxis pushed to the limits the use of inheritance, *meta-classes* become essential when moving to the creation of an **IS software development environments**. The result was the language Telos [17], which allowed one to define (conceptual) models, and to state meta-data constraints. For example, consider the use of a *property category* like "initial condition" in

```
PERSON
  initial condition
     age : {0}
```

This allowed what were essentially assertions in a complex temporal logic to be abbreviated so the time aspects did not appear in formulas. By thinking of properties, such as `age`, as classes (instantiated potentially for each individual of their domain class), property categories, like `initial condition`, become meta-classes, like `INITIAL_PROP_CLASS`: classes with classes as instances. So, given that property classes p have associated $p.Class, p.Name$, and $p.Range$, in our example we would have

`p0.Class=PERSON, p0.Name = age , p0.Range = {0}`

and p0 would be made an instance of property meta-class `INITIAL_PROP_CLASS`, which would be defined to have an (obscure) invariant constraint

$$\forall p \in INITIAL\_PROP\_CLASS.\forall t \in TIME.\forall x.$$
$$(x \in p.Class@t \wedge x \notin p.Class@(t-1)) \Rightarrow x.(p.Name)@t \ \in p.Range$$

describing the temporal semantics.

Incidentally, according to its formalization, a Telos KB consists of a set of "units", each with 4 associated fields: [[source, identifier, destination, time-interval]]. For example,

```
p1 with [[Jimbo,instanceOf,PERSON,2/feb/1992 - 31/dec/2067]]
p2 with [[PERSON,age,{0},all_time]]
p3 with [[p2,instanceOf,INITIAL_PROP_CLASS,all_time]]
Jimbo with [[Jimbo,Jimbo,Jimbo, 2/feb/1992 - 31/dec/2067]]
```

As the last line above indicates, even objects were units with 4 fields.

KR&R Resonances

- RDF(S) anyone? Telos' treatment of properties as objects that link a source to a destination via a name echo RDF's <subject, predicate, object> triples, especially the fact that properties can have properties themselves. The main difference is Telos' addition of a temporal interval.
- While Telos was a theoretical language, it was made into a practical system, ConceptBase [12], by adding deductive rules to it. The considerable success of ConceptBase (downloaded at over 500 sites) lies at least in part in its uniform treatment of everything as an object that can have properties, which allows meta-statements to be easily recorded.

4 First Order Logic

Integrity Constraints are intended to detect inconsistencies after database updates. But, when dealing with the natural world, these constraints are almost always over-generalizations. Hence, one desires to **allow the coexistence of general constraints** such as

$$IC : \forall e.\ e \in EMP \Rightarrow (e.salary > 1000) \wedge (e.salary < e.manager.salary)$$

and occasional exceptions such as

```
calvin.salary=20000
calvin.manager = hobbes
hobbes.salary=15000
```

where the problem might be blamed on the fact that `hobbes` is only temporarily assigned to be `calvin`'s manager. But this leads to a difficulty: once even *one* exception is allowed to persist, this IC will always evaluate to false, and can no longer detect errors in future updates (e.g., when judy's salary is changed to 900), which is "a new reason for it to be false". Therefore we want to modify the constraint to restore its error detection role. In [3], I propose that one first rewrite IC in FOL without function symbols:

$$\forall e.(e \in EMP \wedge sal(e, se)) \Rightarrow (se > 1000) \wedge \\ \forall me, sm.(mgr(e, me) \wedge sal(me, sm)) \Rightarrow se < me)$$

Then, rather than going for the obvious

$$\forall e.(e \neq hobbes \wedge e \in EMP \wedge sal(e, se)) \Rightarrow (se > 1000) \wedge \\ \forall me, sm.(mgr(e, me) \wedge sal(me, sm)) \Rightarrow se < me)$$

(which is not good enough because in this IC there are two conditions that are being checked at the same time), we propose to use the the more subtle

$$\forall e.(e \in EMP \wedge sal(e, se)) \Rightarrow (se > 1000) \wedge \\ ([mgr(e, me) \wedge \neg(e = calvin \wedge me = hobbes)] \wedge sal(me, sm)) \Rightarrow se < me)$$

Interestingly, this corresponds to a *model theoretic* specification: minimally mutilate all models of the original IC, so that the exceptional fact ($manager(calvin, hobbes)$) is counter-factual. The nice property of this is that (i) there is a syntactic transformation corresponding to it, and (ii) the actual syntactic form of the ICs does not matter. (I owe Ray Reiter a debt for helping me see the above.)

KR&R Resonances

My student, Mukesh Dalal used this idea of minimal mutilation of models to obtain a propositional knowledge-base revision operator $update(kb, u)$ [8] — (almost) the first one that was insensitive to the syntactic form of the theory kb or the update u. (To get the models of $update(kb, u)$, it iteratively mutilated single atoms of kb models till at least one model consistent with u was found.) Interestingly, through the advice of David Israel, this seems to also have been the first AI paper to mention the Alchourrón - Gärdenfors - Makinson belief revision axioms.

5 The Classic Description Logic

Classic [5] was probably the most widely used second-generation description logic system: one with precise semantics, polynomial time subsumption checking, and complete algorithm (when `one-of` and `fills` were used with individuals that themselves had no properties, such as numbers, enumerations, etc.).

More significantly, Classic was used in real, industrial applications dealing with configuration management, as well as (attempted) support for data exploration/mining carried out by humans.

Not only was Classic first published in the SIGMOD database conference (rather than an AI conference), but **special strengths of description logics became apparent when viewing DLs as languages used for interacting with a database-like system**. Specifically, one can adopt a Levesque- and SQL-inspired view of a knowledge base management system as a black box with

– *create* operator to declare new identifiers, with possible definitions
– *constrain* operator to express integrity constraints on valid states of the KB
– *update* operator to manage facts about a specific world (A-box)
– *inquire* operator to ask about the state of the world (A-box)

Now, each of the first three operators x above involves a language L_x, while *inquire* involves two languages: one for stating questions, L_{query}, and one for expressing answers, L_{answer}. By considering what happens when each of these languages is a DL such as Classic, one gets the following insights:

- L_{create}: DLs provide the (well-known) opportunity to add not just primitive but also defined concepts ("views"), and to automatically organize these in subsumption hierarchies.
- $\mathrm{L}_{constrain}$: DLs allow necessary conditions to be stated on primitive concepts, which are like Integrity Constraints. (These are the first key ingredient in configuration management applications.)
- L_{update}: By asserting a description such as $\forall friends.(\forall gender.FEMALE)$ about *DonJuan*, one is able to say things about an indefinite number of objects – the current and any yet to be specified friends of Don Juan. This is the source of considerable expressive power for DL-based knowledge/data management, in contrast to null values in relational databases, and the second key to the success of Classic in configuration management. Note that this benefit of dealing with incomplete information accrues even when the language does not support disjunction, and has polynomial time reasoning.
- L_{query}: Of course, DL concepts are well suited to retrieve sets of values — their instances; a benefit here is that queries can themselves be organized in subsumption hierarchies, which facilitates re-use and query refinement. (On the down side, DL concepts cannot express even some very simple conjunctive queries [4] — the bread and butter of database research.)
- L_{answer}: When using DL concepts as part of answers, one gets the benefit of *descriptive*, rather than just enumerated, answers. For example, in response to the query "Who is female?" one might get not just Eve, but also "the friends of Don Juan".

KR&R Resonances

It may be worth recording that essential to the practical success of Classic was the ability to extend its expressive power as needed (through test functions). I believe that OWL has not yet met the hard-nosed challenge of real applications, where customers may walk away if their needs are not served. I do not see any reasons why OWL cannot be made extensible at least to the point of allowing arbitrary "concrete domains", and then maintaining libraries of such extensions, just like we will maintain libraries of concepts (ontologies).

A more interesting reverberation was our application of Classic-like ideas to the specification of Corba services [6], which is exactly like the use of DLs for specifying services on the Semantic Web.

6 Goals

In the past decades, research in Requirements Engineering for software has undergone a revolution, whereby the standard functional specification stage is now preceded by a new phase of *early requirements*, dealing with the intentions of the agents and organizations in the environment where the software is to be used. Because of its focus on goals, and how they are to be achieved, this is known as Goal-Oriented Requirements Engineering (GORE). One of the important aspects of GORE is dealing with so-called non-functional (soft) goals, such as efficiency, accuracy, etc. — goals that do not have clear criteria of success.

The field of database specification, on the other hand, has remained pretty-well unchanged since the mid-70's: one still starts by constructing a conceptual schema as an ER or maybe UML diagram. In recent joint work with Jiang, Topaloglou and Mylopoulos [13], we have started to look at a **goal-oriented approach to database design**. As in GORE, we start with various stakeholders, and their general hard and soft goals; decompose these into subgoals using AND/OR graphs; then apply means/ends analysis to find tasks that achieve them. See Fig. 2 for a small example, which uses the i^* notation [19]. In diagrams, we use contribution edges labeled with + or − (or

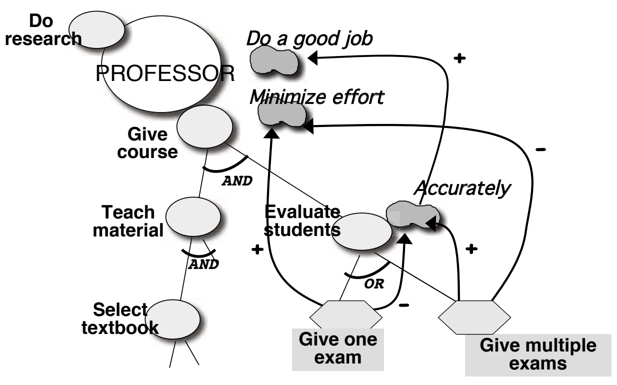

Fig. 2. A Goal Model in i^*

even ++ and −−) to indicate how goals influence each other. Thus, in Fig.2, the softgoal (peanut-shape) of evaluating students accurately contributes positively towards the "Do a good job" softgoal, but performing the task of giving a single exam (hexagon) contributes negatively towards accurate evaluation. One important advantage of goal-oriented approaches is that they provide consideration of design alternatives, and traceability for decisions based on such contribution dependencies. So, for example, we might choose between schemas `StudentTable1(studentId,examGrade)` and `StudentTable1(studentId,exam1, exame2,averageGrade))` based on which goals are more important.

Continuing with our goal-based schema design, our methodology suggests analyzing the textual description of goals, and the participants in the tasks, to obtain a list of "relevant concepts", which is then organized into a *domain model* (expressed in some conceptual modeling language); its purpose is to provide a shared understanding of the domain for database designers and end-users. The *conceptual schema* of the database is then derived from this by addressing a list of questions concerning issues such as persistence, time, data quality, etc. For each such category, we have a number of alternative schema manipulation operators which can be applied to derive the final conceptual schema from the domain model.

KR&R Resonances

Of course, one of the earliest KR&R proposals in Artificial Intelligence was Simon and Newell's GPS, which was also concerned with means-ends analysis to achieve goals. And, in a separate context, it was Simon who introduced the important notion of *"satisficing"*, which is applicable for such goals.

Although it is clear how to formalize AND/OR goal decomposition even in Horn propositional logic, notions like softgoals, their satisficing, and contribution edges would seem to be inherently "soft" — hard to reason with. Sebastiani et al [18] however show how to formalize even this aspect: Since there could be conflicting evidence concerning any goal g, the secret is to replace proposition g by propositions $fully\_satisfied\_g, partially\_satisfied\_g, partially\_denied\_g$, and $fully\_denied\_g$. An edge $g \xrightarrow{-} h$ then introduces axiom

$$partially\_satisfied\_g \Rightarrow partially\_denied\_h$$

while edge $g \xrightarrow{--} h$ also adds axiom

$$fully\_satisfied\_g \Rightarrow fully\_denied\_h$$

By using a suitable extension of this set of axioms, and a min-sat solver, it is then possible to find minimal sets of "input/bottom" goals that guarantee desired top-level goals.

7 Conclusions

I have briefly reviewed a sample of database-inspired projects which had connections to KR&R topics: natural language access to databases ↔ semantic networks; information system design ↔ inheritance & metaclasses; exceptions to database integrity constraints ↔ First Order Logic and minimal mutilations; querying and verifying consistency of incomplete databases ↔ description logics; goal-based database design ↔ goal analysis and satisfaction/satisficing. In each case, I tried to give an impression of the benefits each field, DB and KR&R, derived from the other, as a result of the research carried out. Of course, the above survey was highly skewed towards my own experiences — there are many more such examples, both extant and to come.

Acknowledgments

I am most grateful to all my collaborators over the years for making this such an enjoyable adventure. I single out Enrico Franconi for particular thanks for suggesting "KR&DB" as the topic of this paper, and for his comments on the write up. All remaining errors are of course my own.

This work was supported in part by the U.S. DHS under ONR grant N00014-07-1-0150.

References

1. J.L. Barron, "Dialogue and Process Design for Interactive Information Systems using Taxis", Proceedings ACM SIGOA, pp. 12-20, Philadelphia, June 1982
2. Alexander Borgida: On the Definition of Specialization Hierarchies for Procedures. IJCAI 1981: 254-256
3. Alexander Borgida: Language Features for Flexible Handling of Exceptions in Information Systems. ACM Trans. Database Syst. 10(4): 565-603 (1985)
4. Alexander Borgida: On the Relative Expressiveness of Description Logics and Predicate Logics. Artif. Intell. 82(1-2): 353-367 (1996)
5. Alexander Borgida, Ronald J. Brachman, Deborah L. McGuinness, Lori Alperin Resnick: CLASSIC: A Structural Data Model for Objects. SIGMOD Conference 1989: 58-67
6. Alexander Borgida, Premkumar T. Devanbu: Adding more "DL" to IDL: Towards More Knowledgeable Component Inter-Operability. ICSE 1999: 378-387
7. Chaudhri, V., Hadzilacos, V. and Mylopoulos, J., "Concurrency Control for Knowledge Bases", Third International Conference on Knowledge Representation and Reasoning, Boston, October 1992.
8. Mukesh Dalal: Investigations into a Theory of Knowledge Base Revision. AAAI 1988: 475-479
9. Enrico Franconi, Gary Ng: The i.com tool for Intelligent Conceptual Modeling. KRDB 2000: 45-53
10. Aldo Gangemi, Nicola Guarino, Claudio Masolo, Alessandro Oltramari, Luc Schneider: Sweetening Ontologies with DOLCE. EKAW 2002: 166-181
11. Antoon Goderis, Ulrike Sattler, Carole A. Goble: Applying Description Logics for Workflow Reuse and Repurposing. Description Logics 2005
12. Matthias Jarke, Rainer Gallersdrfer, Manfred A. Jeusfeld, Martin Staudt: ConceptBase - A Deductive Object Base for Meta Data Management. J. Intell. Inf. Syst. 4(2): 167-192 (1995)
13. Lei Jiang, Thodoros Topaloglou, Alexander Borgida, John Mylopoulos: Incorporating Goal Analysis in Database Design: A Case Study from Biological Data Management. RE 2006: 196-204
14. John Mylopoulos, Alexander Borgida, P. Cohen, Nick Roussopoulos, John K. Tsotsos, Harry K. T. Wong: TORUS - A Natural Language Understanding System For Data Management. IJCAI 1975: 414-421
15. John Mylopoulos, P. Cohen, Alexander Borgida, L. Sugar: Semantic Networks and the Generation of Context. IJCAI 1975: 134-142
16. John Mylopoulos, Philip A. Bernstein, Harry K. T. Wong: A Language Facility for Designing Database-Intensive Applications. SIGMOD 1978 (Abstract). ACM Trans. Database Syst. 5(2): 185-207 (1980)

17. John Mylopoulos, Alexander Borgida, Matthias Jarke, Manolis Koubarakis: Telos: Representing Knowledge About Information Systems. ACM Trans. Inf. Syst. 8(4): 325-362 (1990)
18. Roberto Sebastiani, Paolo Giorgini, John Mylopoulos: Simple and Minimum-Cost Satisfiability for Goal Models. CAiSE 2004: 20-35
19. Eric S. K. Yu, John Mylopoulos: From E-R to A-R — Modelling Strategic Actor Relationships for Business Process Reengineering. Int. J. Cooperative Inf. Syst. 4(2-3): 125-144 (1995)