DOOD and DL – Do We Need an Integration?

Paul-Th. Kandzia   Christian Schlepphorst*

Institut für Informatik, Universität Freiburg i. Br., Germany
{kandzia,schlepph}@informatik.uni-freiburg.de

*supported by the Deutsche Forschungsgemeinschaft, La 598/3-2

Abstract

Description logics and deductive object oriented databases provide a similar view on the structure of an application domain, but rely on quite different reasoning paradigms. Recently their combination is dicussed. However, such a general solution is not at hand yet. Fortunately, a thorough investigation of the application and a thoughtful decision for an advanced system using the appropriate of both reasoning paradigms may also lead to a concise and simple implementation, even in cases which at first sight seem not to fit in the declarative framework. Here we describe our experiences with a linguistics problem.

1 Introduction

Over the last years the relationship between advanced database models and description logics (DL) has been increasingly discussed. On the database side deductive object oriented database (DOOD) languages added to traditional database techniques a rich structure as well as rules to describe intensional information. On the other side applications of DL have to handle large amounts of data. Obviously there are strong similarities between both paradigms: DOOD and DL are both derived from first order logic and both arrange information along classification hierarchies. However, the respective reasoning capabilities are quite different.

The essence of DOOD is logic programming, that is the computation of a distinguished model and answering queries in it, based on closed world semantics. In contrast, the core inference of a DL is deciding the satisfiability of a given formula, thus reasoning about all models using an open world approach. Hence, it is a commonplace that the reasoning possibilities of DOOD are limited compared to that of DLs, and that complementary, due to the complex inference, performance of a DL becomes a problem when large amounts of data are involved [Bre94, Sch95]. Particularly, deducing new individuals and relations between them in the A-box is very restricted, what in turn is the main task of a deductive database.

The complementary characteristics of DOOD and DLs have stimulated a new research direction aiming to *combine* the best of both worlds. CARIN [LR96a] is a recent approach extending a restricted DL by horn clauses. The DL is used to define and maintain a concept hierarchy whereas the rules can reason about instances of these concepts. It turns out that the combined inference mechanism is more powerful than DL or horn clauses alone, in particular it can apply rules to incomplete data by dealing with existentially quantified constraints. On the other hand, both sides have to be severely restricted to keep the inference decidable. Admitting recursive horn clauses (what is the main motivation for the deductive approach) leads to undecidability for most non-trivial constructor sets [LR96b]. Therefore, it is questionable whether the CARIN approach is of great help if the user really wants to exploit the power of rule-based reasoning. CARIN also excludes function symbols which are an important feature of DOOD; adding other useful object oriented features to the rule part (complex syntax, closure axioms etc.) would make things even more difficult and is probably not feasible.

In the case of $\mathcal{CVL}$ [CGL95], a DL especially developed for database reasoning, it is not possible to speak about database instances directly. One can define views as combinations of other concepts, for example the transitive closure of a role (e.g. the role `ancestor`), but not apply such a transitive closure to individuals of the A-box (e.g. ask for all ancestors of

Proceedings of the 4th KRDB Workshop Athens, Greece, 30-August-1997

(F. Baader, M.A. Jeusfeld, W. Nutt, eds.)

http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-8/

a certain John). Thus, $\mathcal{CVL}$ is a powerful language to reason about *schemata*, but not to access and query *data*.

However, we are convinced that in many applications such a general integrated system is not really necessary, rather advanced DOOD and DL languages should alternatively be considered. The decision for one of both paradigms should rely on a thorough analysis of the problem and its particular demands, since a choice based on first sight criteria or mere "tradition" may lead to unsatisfying solutions. This proposition is supported by several promising proposals to tackle classical candidates for the database realm like heterogenous databases or schema design with a specialized DL [LR96c]. Our work crosses the border from the other side: An example of the knowledge aquisition area, formerly implemented in a DL system (LOOM), was adapted in the DOOD language F-logic [KLW95] and run on the Florid system [FHK+97, Kan97]. Taking advantage from the distinctive flexibility of F-logic which enables reasoning about data and meta-data in a uniform way – in particular variables can range over objects as well as over classes and methods – the result was a far more concise and clear formulation of the algorithm.

In the following we will first summarize our experiences and then go into some details for illustration.

2 Experiences

A project of the Computational Linguistics Lab of the University of Freiburg (CLIF) approaches the problem of ambiguities in text parsing using a logical qualification calculus [HKS96]. Basically, hypotheses about the item in question are generated due to local knowledge of the text and refined in course of further parsing. Finally, after reading a whole text passage, the most probable hypothesis is chosen by an assessment step, and all others are discarded. Since in this calculus reasoning is about concepts, and since for natural language processing DLs are common use, a prototype was implemented with the LOOM system [Mac94].

However, the resulting formulation of the calculus is not satisfying. Especially the following drawbacks are important:

- The essential knowledge to be modelled is how to obtain new or more exact hypotheses and the definition of criteria for the assessment. This amounts to updates of the A-box and therefore had to be programmed outside of the DL.
- Consequently, there is no need to maintain a concept taxonomy under open world semantics. Hence, the specific reasoning capabilities of a DL are not required.
- Reasoning concerns hypotheses about the text as well as items of the text. For that aim a complex reification framework is needed. Obviously, the data structures provided by the DL were not sufficient for this kind of meta reasoning.

To draw a conclusion, one should not be misled by the consideration that for reasoning about concepts the appropriate means is always a "concept language". Indeed, DLs are not the only languages speaking about concepts, as the notion is quite similar (from a model theoretic point of view) to classes in object oriented modeling. Hence, when chosing the appropriate programming paradigm we are convinced that the following criteria are of much more use:

- What kind of reasoning is required? In our example, the calculus is organized as if–then rules. Furthermore, all derivable consequences are needed for the assessment step. Also, for the calculus there is no difference between open and closed world semantics. These considerations naturally lead to forward chaining.
- What data structures are needed? Here, a concise modeling of text and meta statements on a uniform level is essential.
- How much manipulation of facts (A-box reasoning) is required? Since the new hypotheses have to be stored a lot of new facts is generated.

Discussing the above questions we came to the conclusion that an advanced DOOD system might be a far better choice for implementing the qualification calculus. To support this claim, in the following section we present some details of the CLIF approach and give an overview of a small example taken from [HKS96], using Florid [FHK+97, Kan97] (URL: `http://www.informatik.uni-freiburg.de/~dbis/flogic-project.html`) as a powerful DOOD system.

3 Details of the CLIF example

The algorithm from [HKS96] consists of four major steps. First, the parser reads some text portion and compares it with the knowledge base. If an unknown item is found, it generates a number of hypotheses. Then the hypotheses are **elaborated** by drawing all inferences from them using rules (incorporating linguistic knowledge) and A-box classification mechanisms. In the **Assessment** step, the Qualification Calculus looks for hints for plausibility or implausibility of the hypotheses and assigns *quality labels* to the interesting statements. Finally, **Ranking of Hypotheses** is done by collecting all quality labels for each hypothesis. The hypotheses with the most positive labels and the least negative are considered the

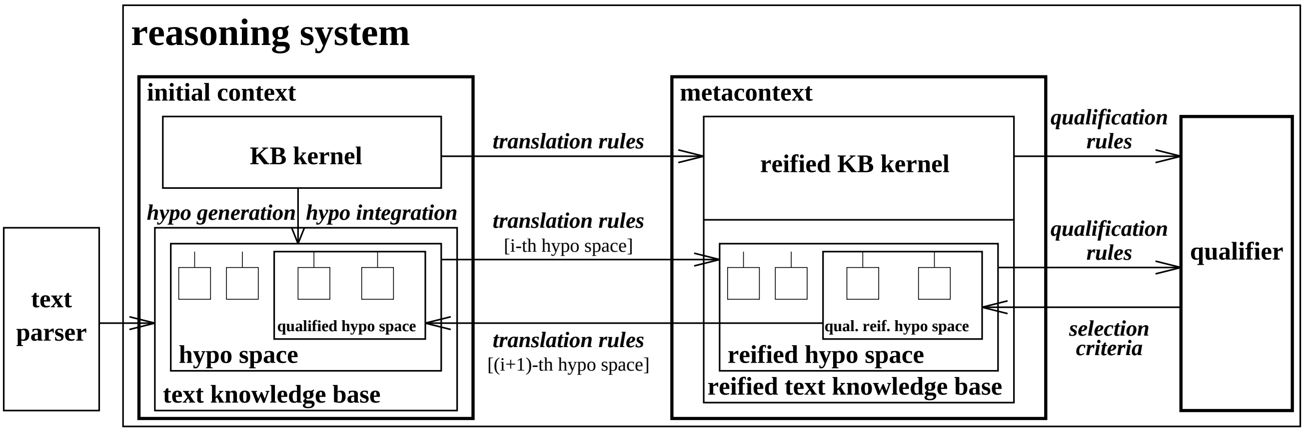

Figure 1: The original architecture from [HKS96]

most convincing. Bad hypotheses, especially inconsistent ones, are discarded. The whole process is repeated in a cycle.

The main contribution of [HKS96] lies in the assessment step, where linguistic knowledge is expressed as if-then rules assigning quality labels. In the context of the example from [HKS96] the following labels are used:

Inconsistency:

If a hypothesis is inconsistent *in itself*, it has to be discarded.

Supported, Cross-Supported:

A statement may be supported by an analogous one. Having many statements supporting each other raises the inner coherence of a hypothesis and thus the confidence. A situation where the *Supported* rule applies looks as follows:

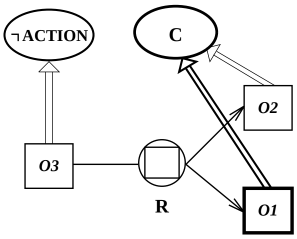

Figure 2: SUPPORT Rule

If *o1* is the unknown item and our hypothesis says, *o1* is a computer, *o3* sells it, and *o3* sells also the computer *o2*, this sounds reasonable.

Additional-Role-Filler:

If an action verb from the source text refers to more than one grammatical object, this lowers the confidence of the hypothesis. However, as such cases are possible, it is not totally discarded. Strong positive quality labels can balance it out.

Multiply-Deduced.

If a statement is true under more than one hypothesis, it is more plausible than others and therefore raises the confidence of all hypotheses supporting it.

The qualification rules reason about statements as well as about hypotheses and thus work on a meta-level. To do this in LOOM, the CLIF group had to build a complex framework for meta-reasoning:

- **Reification.** Statements must be domain items. Therefore an (external) reification function $\Re$ is needed, generating a distinct OID and linking the information of the statement to this member of the REIF concept (r is the unique OID):

  $$\Re(x\ R\ y) \equiv (r : \text{REIF} \ \sqcap\ r\ \text{BINARY-REL}\ R\ \sqcap\ r\ \text{DOMAIN}\ x\ \sqcap\ r\ \text{RANGE}\ y\ \sqcap\ r\ \text{HYPO}\ H)$$

  As a shorthand, a projection function π yields the reificator r of a statement: $\pi(\Re(xRy)) := r$.

  So all knowledge extracted from the source text is stored *in the A-box* accessible to the calculus rules. Obviously, this proceeding has several drawbacks:

  - The representation is lengthy and difficult to read.
  - The external reification function is not part of the DL framework.
  - All implications of statements (e.g. due to subclass inclusion) need to be reified independently, leading to redundancy problems (additional frame rules to ensure consistency etc.).

- **Contexts.** To separate these two representations of the knowledge base (reified and unreified), *contexts* in the sense of McCarthy [McC93] are used. These contain the hypotheses as subcontexts (*hypo spaces*).

Figure 1 (taken from [HKS96]) shows the basic structure of this framework.

As an experiment, we modelled the example from [HKS96]. It is a short german phrase from the real-world domain of information technology test reports (*"Marktanalytiker bestätigen, dass Compaq seit Jahren erfolgreich LTE-Lites anbietet und seit kurzem auch Venturas"*) where it is known that Compaq produces the notebook LTE-lite, and the system tries to handle the ambiguous item *venturas*. It may be another product of Compaq or another producer of LTE-lite. Background information about the real-world domain is expressed in the T-box axioms listed in figure 3. Knowledge about items of this domain like Compaq, LTE-lite etc. is contained in the A-box (figure 4), assuming that the system has gathered it before. It is listed in reified representation.

Now, how can F-logic help to avoid the reification and state the deduction rules of the qualification calculus in a concise and readable way?

- **Hypotheses are objects.** To be able to talk about different hypotheses, we represent them by objects. All such objects are collected in the class hypothesis.

- **Hypotheses form a tree.** New hypotheses are created by refining old ones, enriching them by new statements. To an old hypothesis there may be several sons, i.e. different sets of statements extending the set associated to the old hypothesis. Thus, it is very natural to represent the evolution of hypotheses as a tree. Refinements are *methods* defined for hypotheses that result in derived hypotheses. We can easily write paths from the root to a leaf using *path expressions* [FLU94]:

  h1.r1.r2 : hypothesis

  (A path is built by successively applying methods, here the refinements r1,r2 on the hypothesis h1 using the dot operator.)

- **Domain items are methods on hypotheses.** Using the abstract items of our real world domain as methods operating on hypotheses, we can navigate from a hypothesis to specialized instances representing the abstract object under this hypothesis. Then statements can be formulated directly for the specialized instance:

  h1.compaq[produces$\twoheadrightarrow$venturas],
  h1.venturas : product.

  This means that, under the hypothesis h1, compaq produces venturas, the latter being a product. Again, path expressions are of much help for concise code. Other than in the original version, we do not model hypotheses as a set of *statements* but as a collection of specialized objects representing the text items under this hypothesis. All such objects are linked to the hypothesis object. Statements holding under this hypothesis are expressed using the specialized objects. Nevertheless, this enables generic rules over statements because statements have a fixed format (binary roles).

- **Direct assignment of quality labels to hypotheses.** In the original version, the qualification calculus assigns labels to **statements**. As finally the hypotheses, not the single statements, have to be judged, all labels of a hypothesis are collected and counted. We directly assign the labels to the hypotheses instead. For each kind of label (supp, c_supp, m_ded, add_rf) we have a multivalued method on hypotheses with a signature like the following:

  hypothesis[supp$\Rightarrow\!\!\!\Rightarrow$qualification].

In the above framework the qualification rules can be expressed directly in a readable and concise way, e.g. the SUPPORT rule, where H is a variable ranging over hypotheses (q is used as a Skolem function generating a new object name for the quality label):

```
H[supp->>q(O3,R,O1,O2,C)] :-
      H.O3[R ->>{O1,O2}], not O1=O2,
      H.O1 : C, H.O2 : C,
      not H.O3 : action.
```

Figure 5: SUPPORT Rule

It is important to observe that in this modeling, T-box definitions (representing background knowledge about the real-world domain) are transferred to hierarchy facts, rules and integrity constraints, which leads to a different view on the problem and the algorithm. In the given case, this seemed to be appropriate. In the way sketched above, the whole example was translated to F-logic and executed with the FLORID prototype.

In our eyes, the F-logic version (figure 6) of the A-box facts (figure 4) is a striking proof that a powerful

$$
\begin{array}{lcl}
\textsc{Product} & \doteq & \textsc{Physical-Object} \sqcap \forall\textsc{has-developer}.\textsc{Producer} \\
\textsc{Hardware} & \doteq & \textsc{Product} \sqcap \forall\textsc{has-weight}.\textsc{Weight} \\
\textsc{Notebook} & \doteq & \textsc{Hardware} \sqcap \forall\textsc{has-display}.\text{LCD-}\textsc{Display} \\
\textsc{Accu} & \doteq & \textsc{Hardware} \sqcap \forall\textsc{energy-for}.\textsc{Notebook} \\
\textsc{Company} & \doteq & \textsc{Physical-Object} \sqcap \forall\textsc{produces}.\textsc{Product} \\
\textsc{Producer} & \doteq & \textsc{Company} \sqcap \forall\textsc{produces}.\textsc{Hardware} \\
\multicolumn{3}{l}{\textsc{Notebook-Producer} \doteq \textsc{Producer} \sqcap \forall\textsc{produces}.(\textsc{Notebook} \sqcup \textsc{Accu})} \\
\textsc{Offer} & \sqsubseteq & \textsc{Action} \sqcap \forall\textsc{agent}.\textsc{Producer} \sqcap \forall\textsc{patient}.\textsc{Product} \\
\textsc{Develop} & \sqsubseteq & \textsc{Action} \sqcap \forall\textsc{agent}.\textsc{Producer} \sqcap \forall\textsc{patient}.\textsc{Hardware}
\end{array}
$$

Figure 3: Original T-box axioms for the Compaq example

$$
\begin{array}{lcl}
p_1 & = & \pi(\Re(\mathit{Compaq} : \textsc{Notebook-Producer})) \sqcap p_1 \text{ QUALIFIED } q_1 \\
p_2 & = & \pi(\Re(\mathit{Compaq} : \textsc{Producer})) \sqcap p_2 \text{ QUALIFIED } q_2 \\
p_3 & = & \pi(\Re(\mathit{Compaq} : \textsc{Company})) \sqcap p_3 \text{ QUALIFIED } q_3 \\
p_4 & = & \pi(\Re(\mathit{Compaq} : \textsc{Physical-Object})) \sqcap p_4 \text{ QUALIFIED } q_4 \\
p_5 & = & \pi(\Re(\mathit{Compaq} \text{ PRODUCES } \mathit{LTE\text{-}Lite})) \sqcap p_5 \text{ QUALIFIED } q_5 \\
p_6 & = & \pi(\Re(\mathit{LTE\text{-}Lite} \text{ HAS-DEVELOPER } \mathit{Compaq})) \sqcap p_6 \text{ QUALIFIED } q_6 \\
p_7 & = & \pi(\Re(\mathit{LTE\text{-}Lite} : \textsc{Notebook})) \sqcap p_7 \text{ QUALIFIED } q_7 \\
p_8 & = & \pi(\Re(\mathit{LTE\text{-}Lite} : \textsc{Hardware})) \sqcap p_8 \text{ QUALIFIED } q_8 \\
p_9 & = & \pi(\Re(\mathit{LTE\text{-}Lite} : \textsc{Product})) \sqcap p_9 \text{ QUALIFIED } q_9 \\
p_{10} & = & \pi(\Re(\mathit{LTE\text{-}Lite} : \textsc{Physical-Object})) \sqcap p_{10} \text{ QUALIFIED } q_{10} \\
p_{11} & = & \pi(\Re(\mathit{NiMH\text{-}Accu} \text{ ENERGY-FOR } \mathit{LTE\text{-}Lite})) \sqcap p_{11} \text{ QUALIFIED } q_{11} \\
p_{12} & = & \pi(\Re(\mathit{NiMH\text{-}Accu} : \textsc{Accu})) \sqcap p_{12} \text{ QUALIFIED } q_{12} \\
p_{13} & = & \pi(\Re(\mathit{NiMH\text{-}Accu} : \textsc{Hardware})) \sqcap p_{13} \text{ QUALIFIED } q_{13} \\
p_{14} & = & \pi(\Re(\mathit{NiMH\text{-}Accu} : \textsc{Product})) \sqcap p_{14} \text{ QUALIFIED } q_{14} \\
p_{15} & = & \pi(\Re(\mathit{NiMH\text{-}Accu} : \textsc{Physical-Object})) \sqcap p_{15} \text{ QUALIFIED } q_{15}
\end{array}
$$

Figure 4: Original A-box knowledge for the Compaq example

```
root:hypothesis.

root.compaq:nb_producer[produces->>lte_lite].
root.lte_lite:notebook[has_developer->>compaq].
root.nimh_accu:accu[energy_for->>lte_lite].
```

Figure 6: F-logic formulation of the A-box knowledge of figure 4

DOOD language like F-logic is a better choice for this application.

4 Conclusion

We have seen that at the borderline between knowledge representation and databases both paradigms have their merits and that it is not always immediately clear which one to choose. As our background is in the DOOD field, we were expecially interested in extending the scope of DOOD. The study of a small example from a DL application revealed some major drawbacks. In contrast, its concise DOOD formulation is able to promote the discussion about the linguistic heuristics without distracting by implementation details like the reification in the original version. However, it is noteworthy that to that aim the object oriented features (path expressions, object creation, second order syntax) of an advanced system were necessary.

In no way we claim that DOOD can take the place of DLs but mention the following research topics towards a systematic investigation:

- Characterization of applications for which the DOOD approach is more appropriate than DLs and vice versa. What is the influence of factors like whether the emphasis lies on structure or data, the way to reason, and the sheer *size* of data?
- What are suitable DOOD languages for those applications? Possible candidates are subsets of F-logic (to have maximal performance).

We are convinced that such a discussion will stimulate research and be of use for both communities.

Acknowledgements

Our thanks go to Klemens Schnattinger and Steffen Staab from CLIF; furthermore to Werner Nutt who suggested the experiment.

References

[Bre94] Paolo Bresciani. Uniformly Querying Knowledge Bases and Data Bases. In *KRDB*, 1994.

[CGL95] Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. Structured objects: Modeling and reasoning. In Tok Wang Ling, Alberto O. Mendelzon, and Laurent Vieille, editors, *Proc. Intl. Conference on Deductive and Object-Oriented Databases (DOOD)*, number 1013 in LNCS, Singapore, 1995. Springer.

[FHK+97] Jürgen Frohn, Rainer Himmeröder, Paul-Th. Kandzia, Georg Lausen, and Christian Schlepphorst. FLORID: A prototype for F–Logic. In *Proc. Intl. Conference on Data Engineering*, 1997.

[FLU94] Jürgen Frohn, Georg Lausen, and Heinz Uphoff. Access to objects by path expressions and rules. In Jorge B. Bocca, Matthias Jarke, and Carlo Zaniolo, editors, *Proc. Intl. Conference on Very Large Data Bases*, Santiago de Chile, 1994.

[HKS96] U. Hahn, M. Klenner, and K. Schnattinger. Automated knowledge acquisition meets metareasoning: Incremental quality assessment of concept hypotheses during text understanding. In *Proc. of 10th Knowledge Acquisition Workshop (KAW'96)*, 1996.

[Kan97] Paul-Th. Kandzia. Nonmonotonic Reasoning in Florid. In *Proc. Intl. Conference on Logic Programming and Nonmonotonic Reasoning*, number 1265 in LNAI. Springer, 1997.

[KLW95] Michael Kifer, Georg Lausen, and James Wu. Logical foundations of object-oriented and frame-based languages. *Journal of the ACM*, 42(4):741–843, July 1995.

[LR96a] Alon Y. Levy and Marie-Christine Rousset. CARIN: A representation language combining horn rules and description logics. In *Proc. European Conference on Artificial Intelligence*, Budapest, 1996.

[LR96b] Alon Y. Levy and Marie-Christine Rousset. The limits on combining recursive horn rules with description logics. In *Proc. Thirteenth Nat. Conference on Artificial Intelligence*, Portland, OR, 1996.

[LR96c] Alon Y. Levy and Marie-Christine Rousset. Using Description Logics to Model and Reason about Views. In *KRDB*, 1996.

[Mac94] R. M. MacGregor. A description classifier for the predicate calculus, 1994.

[McC93] J. McCarthy. Notes on formalizing context. In *Proc. of IJCAI'93*, 1993.

[Sch95] Klaus Schild. The Use of Description Logics as Database Query Languages. In *KRDB*, 1995.