Supporting Information Disclosure in an Evolving Environment

A.H.M. ter Hofstede and H.A. Proper and Th.P. van der Weide

Department of Information Systems, University of Nijmegen
Toernooiveld, NL-6525 ED Nijmegen, The Netherlands
{arthur,erikp,tvdw}@cs.kun.nl

PUBLISHED AS:

A.H.M. ter Hofstede, H.A. Proper, and Th.P. van der Weide. Supporting Information Disclosure in an Evolving Environment. In D. Karagiannis, editor, *Proceedings of the 5th International Conference DEXA'94 on Database and Expert Systems Applications*, volume 856 of *Lecture Notes in Computer Science*, pages 433–444, Athens, Greece, EU, September 1994. Springer Verlag, Berlin, Germany, EU. ISBN 3540584358

Abstract. Even if high-level query languages are used, query formulation may cause problems. This is notably so in case of large and complex application domains. Typical examples of these kinds of application domains are evolving application domains. In an evolving application domain not only populations may change, but also the conceptual schema. Even more, the history of the application domain should be recorded, and be retrievable.
This paper focuses on support for query formulation in the context of large conceptual schemata. The solution presented uses the idea of query-by-navigation in conjunction with query-by-construction. First this idea is illustrated by means of some examples, then it is formally defined.

1 Introduction

Nowadays a large number of organisations use automated information systems. A large body of vital corporate information is stored in these information systems. As important decisions are to be based on this information, a primary function of information systems is the retrieval of stored information. Adequate support for information disclosure is far from a trivial problem, as conceptual schemata of real-life applications tend to be quite large and complicated. This problem becomes even more complicated in the context of *evolving information systems*.

In rapidly changing organisations, the flexibility offered by evolving information systems is indispensable. Evolving information systems support the evolution of *all* aspects of applications models [MS90, BKKK87, KBC+89, PW94]. Traditional information systems only allow the evolution of populations. Application models not only contain populations, but also structural aspects of the Universe of Discourse involved, such as constraints and action specifications (see [PW94]). For information disclosure in evolving information systems, not only the current application model, but also past application models may be of interest. Evidently, this yields extra complications for information disclosure. Traditional query languages such as SQL are inadequate for information disclosure in evolving information systems, as they do not accommodate for changes in the *structure* of the involved information. In some approaches to evolving information systems, manipulation languages for relational schemata are extended with historical operations, both on the instance (population) and the schema level [MS90, BKKK87, KBC+89]. However, as a result of their strong connection with the Relational Model, these languages are not on a conceptual level and consequently, not suited for information disclosure.

The approach followed in this paper is inspired by [Big88, GS90, Hag92, Pro93], in which hypertext browsers for (evolving) information systems are described. A further refinement of the idea of a hypertext browser is a *query-by-navigation* interface, allowing users to formulate queries interactively [BPW93]. This approach is based on an identical approach followed in information retrieval systems [BW92]. Query-by-navigation, described in [BPW93], allows for the formulation of rather simple queries only. In this paper the formulation of more complex queries is supported also. This is achieved by extending *query-by-navigation* with *query-by-construction*.

The idea outlined in this paper is presented in terms of the stratified hypermedia architecture described in [BW92]. This architecture consists of two layers, a descriptive layer, referred to as the *hyperindex*, and an instantiation layer, referred to as the *hyperbase*. The hyperindex can be seen as a hypertext of characterisations. The hyperbase contains the actual information. In this paper the hyperindex corresponds to an information structure, while the hyperbase corresponds to a population of that structure. During the query formulation process, users are guided through the descriptive layer.

The structure of this paper is as follows. In section 2 the notion of evolving information system is illustrated. This section also addresses the concept of disclosure schema. A disclosure schema covers all retrievable information of an evolving information system. The necessity of a query-by-navigation system in the context of evolving information systems is stressed. In section 3 the ideas behind a query-by-navigation/query-by-construction system are discussed by a sample session. Section 4 presents the main ingredients of a formal foundation of such a system. Finally, it should be stressed that the approach described in this paper is a generic one, in the sense that the results can be applied to any object-role based data modelling technique. Examples of object-role based data modelling techniques are ER or EER, NIAM ([NH89]), and PSM ([HW93]).

2 Evolving Information Systems

For evolving information systems, an application model is not restricted to instantiations (populations) of the schema involved. Application models then also describe the object types, constraints, and action specifications involved. A collective noun for these modelling concepts is *application model element*. In an evolving information system each application model element is allowed to change in the course of time.

In this section the concept of evolving information system is illustrated by means of a simple example. Next, the disclosure schema of an evolving information system is discussed.

2.1 Example evolving domain

As an illustration of an evolving Universe of Discourse, consider an insurance company for cars. For each policy sold, the insured car and owner are recorded. For each insured car the registration number and type are recorded. Clients are identified by their name and address. In figure 1 an ER schema of this simple Universe of Discourse is presented. The following conventions are used. Names of attributes are preceded by a hash (#). A name above a line, connecting an entity type with a relationship type, verbalizes how that entity type participates in that relationship type (e.g. Car insured by Policy). A name below such a line verbalizes the reverse connection (e.g. Policy insuring Car).

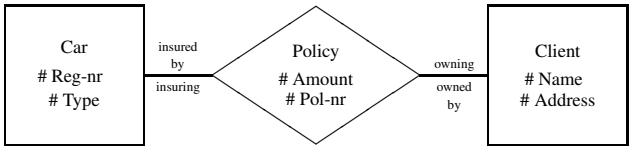

Fig. 1. The information structure of a car insurance company

Suppose that after some time the insurance company notices a substantial difference between damage claims for private cars and those for company cars. Rather than raising the overall policy prices, they effectuate a price differentiation. Policy prices for company cars are raised by some fixed percentage. Policy prices for private cars become dependent on the usage of the car, i.e. the number of kilometers per year. The new ER schema is depicted in figure 2. Entity type Car now has two subtypes, Private car and Company car. Only for Private car the number of kilometers per year is recorded.

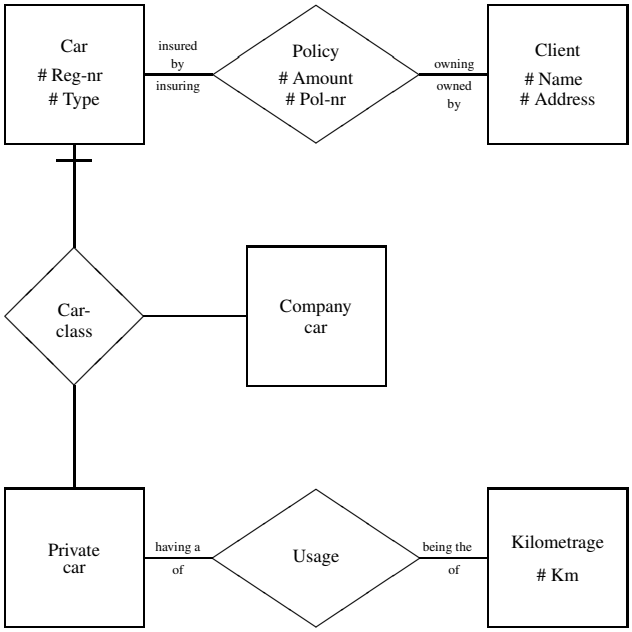

Fig. 2. Car insurance with price differentiation

After some time a number of small companies, not using their cars very intensively, start to protest against the introduced price differentiation. The insurance company then decides to abolish the price differentiation. Instead, a no-claims reduction is introduced. This leads to the attribute Reduction for relationship type Policy. The resulting ER schema is shown in figure 3.

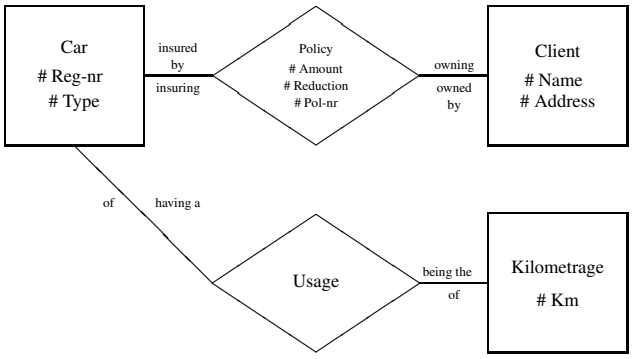

Fig. 3. Car insurance with reduction rates

2.2 Disclosure schema

In our approach, the history of an application domain is captured by the history of its elements. For example, the object type history for policies starts as a relationship type with two attributes, and then evolves to a relationship type with three attributes.

In an evolving information system, not only queries about the current application model can be formulated, but also about past application models. The disclosure schema of an evolving information system is defined as the conceptual schema capturing all information in the evolving information system that can be retrieved. A disclosure schema therefore contains the following components:

Extra-temporal schema The *extra-temporal schema* of an evolving application domain is the union of past and present information structures. In case of the car insurance company, the extra-temporal schema at least comprises the schemata represented in figures 1, 2 and 3.

Meta-schema The meta-schema(ta) of the modelling techniques used for the application model should also be present in the disclosure schema involved, enabling the formulation of queries about structural aspects of evolving information systems. Further, the notion of time, used to describe versions of application models, has to be available in the disclosure schema.

Integration In the disclosure schema, the meta and application level are integrated. Concepts in the application level are linked to their corresponding concepts in the meta level. To this end, the meta-schema and extra-temporal schema are integrated in the disclosure schema via instance of relationships. The meta-schema(ta) of the modelling techniques used for the application model should also be present in the disclosure schema involved.

Evidently, a disclosure schema can easily become very large. Adequate information disclosure support is therefore imperative. In the following section the idea of guided query construction is introduced by a sample session.

3 Guided Query Construction

In this section, we give an idea of the look and feel of the query constructor by a sample session. As sample query language the language LISA-D ([HPW93]) is employed. LISA-D can be seen as a general query language for object-role based data modelling techniques.

Suppose a searcher is interested in insured cars with a usage less than 1000 kilometers per year. In figure 4 the starting screen of the query constructor is depicted. In the middle of the query screen, the current query is depicted. Since the searcher has not yet started, the current focus is Start. After selecting the Start box, a navigation session is initiated, leading to the screen in figure 5,which is referred to as the *starting node* of the navigator.

This screen contains the names of all object types of the disclosure schema under consideration (both object types from the application, and the meta level). The searcher can now proceed by selecting one of the presented items. This selection is the first refinement of the searcher's information need. A selection of an item in a node is denoted by the symbol $\nwarrow$ in the figures. The symbol $\bigtriangledown$ represents a button for a refinement step, while the symbol $\bigtriangleup$ is used for enlargement. Finally, $\triangleright$ represents a button used for related formulations.

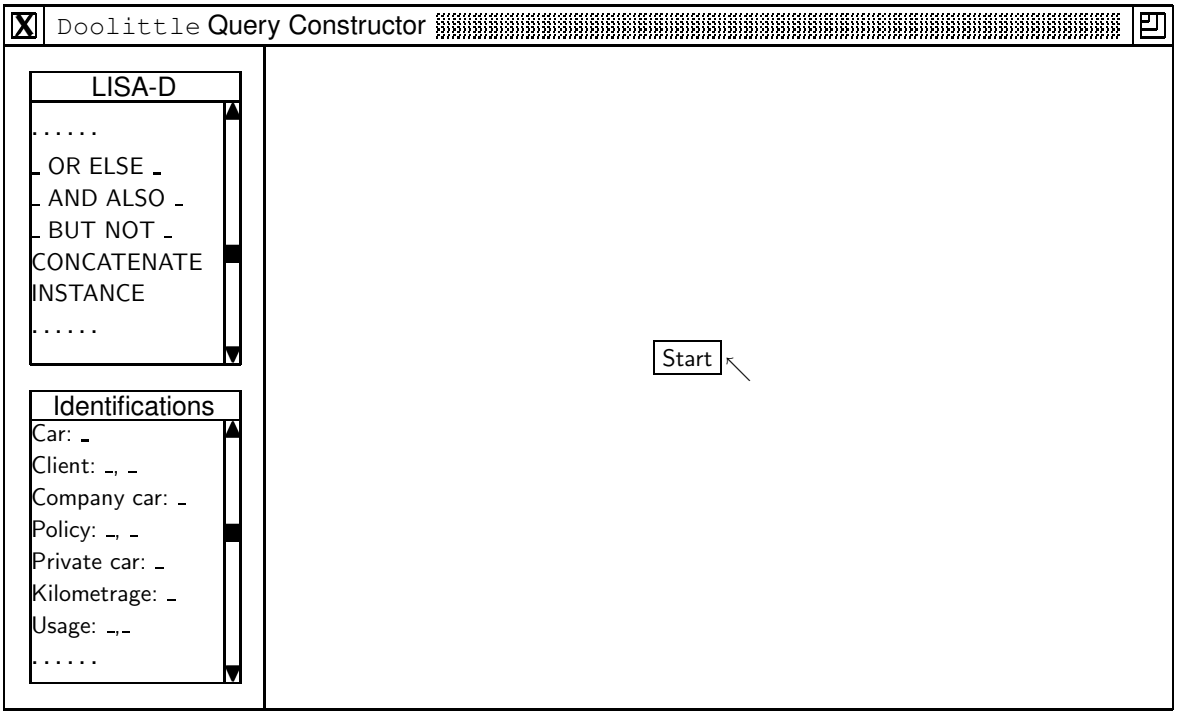

Fig. 4. Opening screen of the query constructor

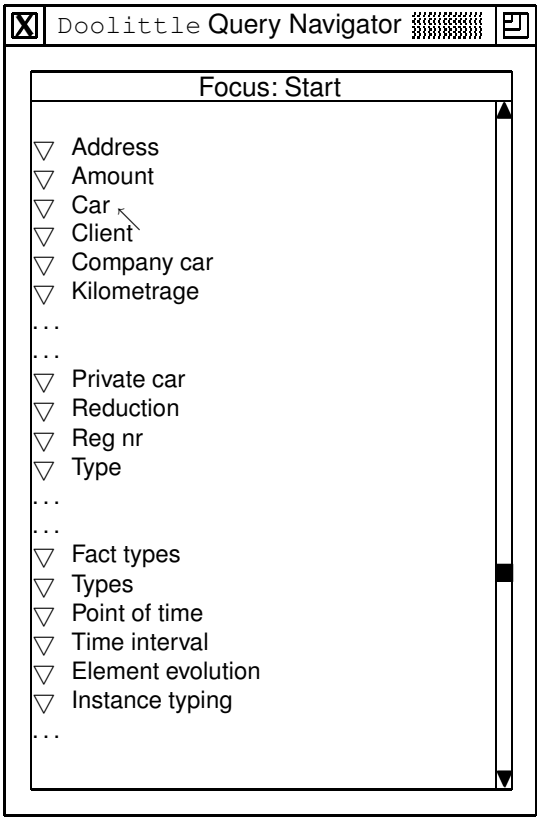

Fig. 5. Selecting a start object type

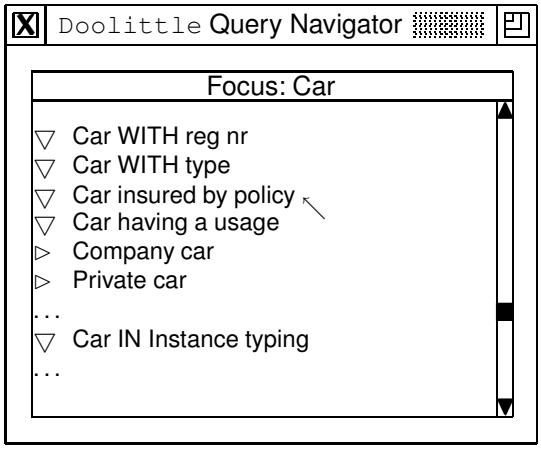

Fig. 6. Starting point: cars

Since the searcher is mainly interested in cars, the item Car is selected. This choice leads to the screen of figure 6. The focus has changed from Start to Car, and the screen depicts all possible continuations for queries starting from Car. Further, Company car and Private car are offered to the searcher as alternative formulations, since they are subtypes of Car. Mostly, alternative formulations originate from subtyping and generalisation. The searcher is interested in insured cars, and consequently selects the item Car insured by policy. This brings the searcher to a next screen, in which the item Car insured by policy owned by client is selected. In the mean time, the query builder screen has changed its focus accordingly.

As the searcher is not interested in all insured cars, but rather those with a usage of less than 1000 kilometers, the searcher continues with selecting the AND ALSO operator from the Elisa-D menu, a binary operator on LISA-D queries. This is an example of *query by construction*. The searcher can now start another navigation session to specify the interest in cars with a usage less than 1000km.

Now suppose that the searcher realises that only cars insured in 1993 are of interest. Therefore, a restriction of the original query is necessary. This can be achieved by a selection of the DURING operation from the Elisa-D menu. A during clause can be put anywhere in a query. As only the policies of 1993 are of interest, the DURING clause is inserted after policy. The resulting screen is shown in figure 7.

4 The Underlying Architecture

In this section a more formal background of the query-by-navigation/query-by-construction system is provided. For the underlying stratified hypermedia architecture, we refer to [BW92]. This architecture offers the possibility to make a separation between several information abstraction levels. In the two level approach, one level (the hyperbase) is concerned with concrete information, the second level (the hyperindex) addresses the structural aspects of the information. In the previous section, it was demonstrated how the hyperindex is used to describe and access information from the hyperbase. We saw how queries can be constructed via a guided tour, using the mechanisms of query-by-construction and query-by-navigation. We restrict ourselves in the rest of this paper to the description of the latter mechanism.

4.1 The Hyperindex

Object role models (such as NIAM, ER, EER, PSM), allow for the verbalization of queries by so-called linear path expressions. A linear path expression describes a (linear) path through the information structure, via object types and predicators (we prefer the term predicator above the term role, see [BHW91]). In

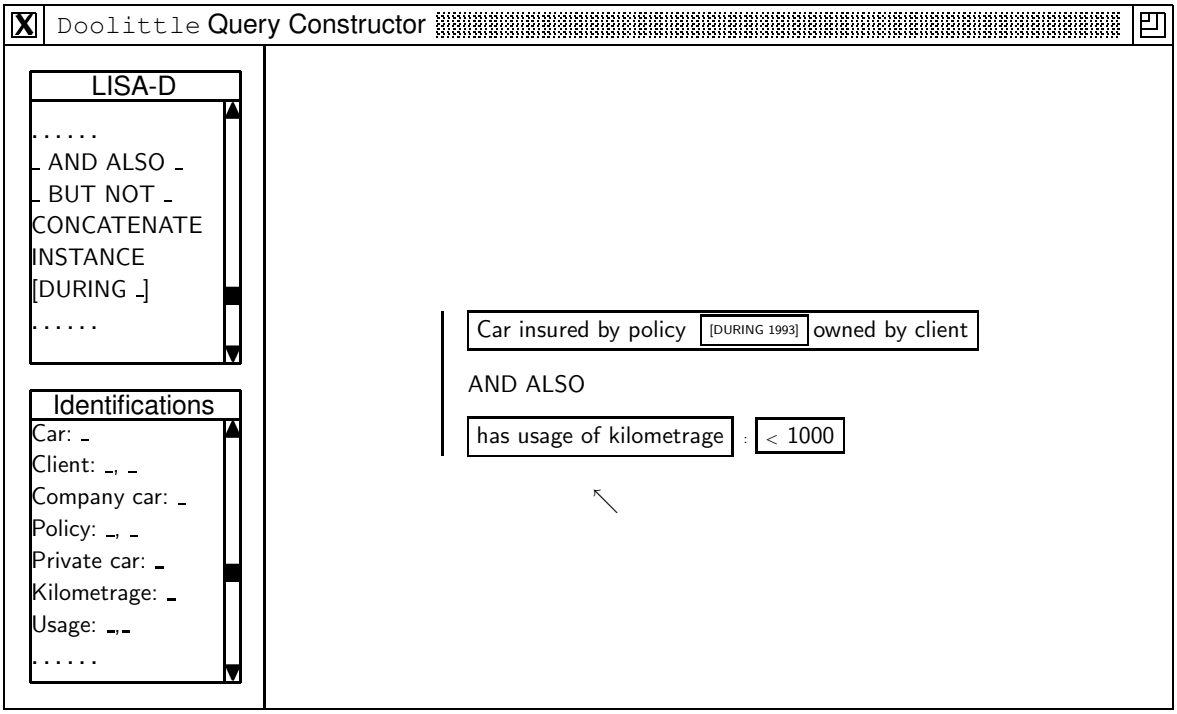

Fig. 7. Query extended with a temporal selection

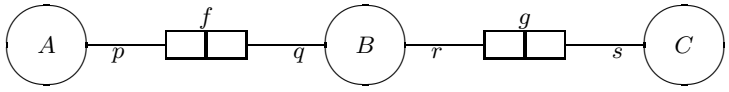

Fig. 8. A small schema with identified predicators

figure 8, A is a path expression which describes a path from A to A. Predicator p is also a valid path expression, representing the path from A (its base) to f (its fact type) via this predicator. The reverse path (from f to A) is denoted as $p^{\leftarrow}$. A more complex example is path expression: $A \circ p \circ f \circ q^{\leftarrow} \circ B \circ r \circ g \circ s^{\leftarrow} \circ C$. This expression corresponds to the path from A to C via f, B and g. Later, we will see how linear path expressions can be verbalized as almost natural language sentences. For more details on (linear) path expressions, refer to: [HPW93].

Linear path expressions will be the molecules for the hyperindex to be constructed. The associated nodes will contain, amongst others, the verbalization of linear path expressions. The components of this hyperindex will be introduced successively.

Fragment base The fragment base ($\mathbb{F}_{\text{ind}}$) of the hyperindex consists of the names for elementary path expressions: object types, predicators and reverse predicators.

Node base Nodes are used to visualize (abstract) molecules, or, linear path expressions. Besides verbalizing the path expression, they present the searcher some context sensitive information, that may help during the process of query formulation.

The structure of a node is as follows. The node heading contains the verbalized path expression $\rho(c)$. The node body consists of three components, a set of less specific path expressions $(u_1, \ldots, u_l)$, a set of

more specific path expressions $(d_1, \ldots, d_m)$, and a set of associated path expressions $(a_1, \ldots, a_n)$. The verbalization function ρ will be introduced at the end of this section.

Schema The simple structure of linear path expressions can be described using a single syntactic category $\langle P \rangle$. The context-free production rules define the way in which linear path expressions can be extended. For any predicator $p \in \mathcal{P}$ and abstract object type $x \in \mathcal{N}$, we have the following rules:

$$\begin{aligned} \langle P \rangle &\rightarrow x \\ \langle P \rangle &\rightarrow \langle P \rangle \circ p \circ \mathsf{Fact}(p) \\ \langle P \rangle &\rightarrow \langle P \rangle \circ p^{\leftarrow} \circ \mathsf{Base}(p) \end{aligned}$$

Views The hyperindex for a disclosure schema, contains only one single view. The start symbol of this view is obviously $\langle P \rangle$. The molecules $\mathbb{M}_{\text{ind}}$ are formed by the set of linear path expressions. The parse trees ω_{ind} describe the construction of these expressions. The presentation of molecules by π_{ind} is covered in the next subsection.

In this view, two associative link schemata are introduced. The first link schema is used to handle specialisation and generalisation occurring in the disclosure schema. If object type x is a supertype of object type y, then with a reference to x may be associated a reference to y. For example, figure 6 contains the link from Car to Company car. This association is, however, only recorded in $\mathbb{L}_{\text{ind}}$ for linear path expressions ending in object type x. The second link schema is used for reversal of the current focus. Normally, a focus is enlarged or refined by operating on the tail of this expression. It may, however, be convenient to be able to operate on the head of the focus also. For this purpose, reversal of the currently focussed path expression can be employed, and is added as an associative link.

Presentation of molecules Molecules (linear path expressions) are presented by nodes that provide a verbalization of the path expression, and an overview of the immediate context of the expression. This presentation is described by the function π_{ind}.

What remains to be done with respect to the presentation of the molecules, is a proper definition of the verbalization function ρ. This is achieved by means of a set of derivation rules, which also derive an associated preference for verbalizations (using penalty points). The predicate $\rho(P, n, x)$ is employed to denote that path expression P is verbalised as n, with penalty x.

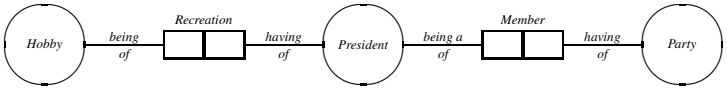

Fig. 9. A naming of figure 8

The name of the empty path expression is defined by the following rule:

[Verb1] (*empty path expression*) $\vdash \rho(\epsilon, \mathsf{Start}, 0)$

Object types and predicators are abstract objects, which are verbalized by associating meaningful names. Object types get a unique name, recorded by the naming function ONm. Predicators have associated a name, which identifies the predicator within its fact type. This predicator name (PNm) also verbalizes the linear path from its base to its associated fact type. Besides, predicators have a reverse name (RNm),

which verbalizes the reversed path, from fact type to base. A possible naming for the schema in figure 8 is provided in figure 9:

$$\begin{array}{lll} \mathsf{ONm}(A) = \mathsf{Hobby} & \mathsf{PNm}(p) = \mathsf{being} & \mathsf{RNm}(p) = \mathsf{of} \\ \mathsf{ONm}(B) = \mathsf{President} & \mathsf{PNm}(q) = \mathsf{having} & \mathsf{RNm}\, q = \mathsf{of} \\ \mathsf{ONm}(C) = \mathsf{Party} & \mathsf{PNm}(r) = \mathsf{being\ a} & \mathsf{RNm}\, r = \mathsf{of} \\ \mathsf{ONm}(f) = \mathsf{Recreation} & \mathsf{PNm}(s) = \mathsf{having} & \mathsf{RNm}\, s = \mathsf{of} \\ \mathsf{ONm}(g) = \mathsf{Member} & & \end{array}$$

As another example, the name 'insuring' in figure 1 refers to the association between the relationship type named 'Policy', and the object type named 'Car'. Predicator names are always placed above lines connecting entity types and relationship types, while reverse predicator names are placed below those lines.

In the derivation of a verbalization for linear path expressions, names provided by the user are exploited as much as possible. This leads to the following derivation rules, assigning no penalties to these names:

[Verb2] (*predicator naming*) $\mathsf{PNm}(p) = n \ \vdash \ \rho(p, n, 0)$

[Verb3] (*reverse predicator naming*) $\mathsf{RNm}(p) = n \ \vdash \ \rho(p^{\leftarrow}, n, 0)$

[Verb4] (*object type naming*) $\mathsf{ONm}(x) = n \ \vdash \ \rho(x, n, 0)$

Verbalization of concatenated path expressions is easily derived via juxtaposition. In order to favour user names assigned to path expressions, the derivation assigns a penalty for each such derivation step.

[Verb5] (*concatenation*) $\rho(P_1, n_1, \alpha_1) \wedge \rho(P_2, n_2, \alpha_2) \ \vdash \ \rho(P_1 \circ P_2, n_1\ n_2, \alpha_1 + \alpha_2 + 1)$

This leads, for the schema in figure 8 with namings as provided by figure 9, to the following verbalizations:

$$\rho(A \circ p \circ f \circ q^{\leftarrow} \circ B, \mathsf{Hobby\ being\ Recreation\ of\ President}, 4)$$
$$\rho(B \circ r \circ g \circ s^{\leftarrow} \circ C, \mathsf{President\ being\ a\ Member\ of\ Party}, 4)$$

The verbalization rules using predicator names and reverse predicator names, sometimes lead to clumsy sentences. Therefore, extra names for transitions via fact types of the form $(p \circ f \circ q^{\leftarrow})$ are usually introduced (by the analyst) in a schema. Such a transition is referred to as a *connector*, and denoted as $\langle p, q \rangle$. In the example of figure 8 we introduce the following connector names:

$$\mathsf{Connector(of)} = \langle p, q \rangle \qquad \mathsf{Connector(being\ member\ of\ )} = \langle r, s \rangle$$

Connector names are used for verbalizations as follows:

[Verb6] (*connector naming*) $\mathsf{Connector}(n) = \langle p, q \rangle \ \vdash \ \rho(p \circ \mathsf{Fact}(p) \circ q^{\leftarrow}, n, 0)$

The verbalizations of the previous example can now be extended with:

$$\rho(A \circ p \circ f \circ q^{\leftarrow} \circ B, \mathsf{Hobby\ of\ President}, 2)$$
$$\rho(B \circ r \circ g \circ s^{\leftarrow} \circ C, \mathsf{President\ being\ member\ of\ Party}, 2)$$

Note the lower penalty value for these verbalizations.

The derivation rules for verbalizations are used to derive a verbalization with highest preference (lowest penalty), as follows.

$$\mathsf{Cost}(P) \triangleq \mathsf{min}\left\{\alpha \ \middle|\ \exists_n \left[\rho(P, n, \alpha)\right]\right\}$$
$$\rho(P) \triangleq n \text{ such that } \rho(P, n, \mathsf{Cost}(P))$$

In our example, we have: $\mathsf{Cost}(A \circ p \circ f \circ q^{\leftarrow} \circ C) = 2$, leading to the following verbalization:

$$\rho(A \circ p \circ f \circ q^{\leftarrow} \circ C) = \mathsf{Hobby\ of\ President}$$

References

[BHW91] P. van Bommel, A.H.M. ter Hofstede, and Th.P. van der Weide. Semantics and verification of object-role models. *Information Systems*, 16(5):471–495, October 1991.

[Big88] J. Bigelow. Hypertext and CASE. *IEEE Software*, 5(2):23–27, 1988.

[BKKK87] J. Banerjee, W. Kim, H.J. Kim, and H.F. Korth. Semantics and Implementation of Schema Evolution in Object-Oriented Databases. *SIGMOD Record*, 16(3):311–322, December 1987.

[BPW93] C.A.J. Burgers, H.A. Proper, and Th.P. van der Weide. Organising an Information System as Stratified Hypermedia. In H.A. Wijshoff, editor, *Proceedings of the Computing Science in the Netherlands Conference*, pages 109–120, Utrecht, The Netherlands, EU, November 1993.

[BW92] P.D. Bruza and Th.P. van der Weide. Stratified Hypermedia Structures for Information Disclosure. *The Computer Journal*, 35(3):208–220, 1992.

[GS90] P.K. Garg and W. Scacchi. A Hypertext System to Manage Software Life-Cycle Documents. *IEEE Software*, 7(3):90–98, 1990.

[Hag92] T.M. Hagensen. Hyperstructure CASE Tools. In B. Theodoulidis and A. Sutcliffe, editors, *Proceedings of the Third Workshop on the Next Generation of CASE Tools*, pages 291–297, Manchester, United Kingdom, May 1992.

[HPW93] A.H.M. ter Hofstede, H.A. Proper, and Th.P. van der Weide. Formal definition of a conceptual language for the description and manipulation of information models. *Information Systems*, 18(7):489–523, October 1993.

[HW93] A.H.M. ter Hofstede and Th.P. van der Weide. Expressiveness in conceptual data modelling. *Data & Knowledge Engineering*, 10(1):65–100, February 1993.

[KBC+89] W. Kim, N. Ballou, H.-T. Chou, J.F. Garza, and D. Woelk. Features of the ORION Object-Oriented Database. In W. Kim and F.H. Lochovsky, editors, *Object-Oriented Concepts, Databases, and Applications*, ACM Press, Frontier Series, pages 251–282. Addison-Wesley, Reading, Massachusetts, 1989.

[MS90] E. McKenzie and R. Snodgrass. Schema evolution and the relational algebra. *Information Systems*, 15(2):207–232, 1990.

[NH89] G.M. Nijssen and T.A. Halpin. *Conceptual Schema and Relational Database Design: a fact oriented approach*. Prentice-Hall, Sydney, Australia, 1989. ASIN 0131672630

[Pro93] H.A. Proper. Towards an Integration of Evolving Information Systems and CASE-Tools. In S. Brinkkemper and F. Harmsen, editors, *Proceedings of the Fourth Workshop on the Next Generation of CASE Tools*, pages 23–33, Paris, France, EU, June 1993. ISSN 09243755

[PW94] H.A. Proper and Th.P. van der Weide. EVORM - A Conceptual Modelling Technique for Evolving Application Domains. *Data & Knowledge Engineering*, 12:313–359, 1994.

This article was processed using the LaTeX macro package with LLNCS style