Computer Supported Query Formulation in an Evolving Context

A.H.M. ter Hofstede[1] and H.A. Proper[2,3] and Th.P. van der Weide[1]

[1]Computing Science Institute
University of Nijmegen
Toernooiveld
NL-6525 ED Nijmegen

[2]Department of Computer Science
University of Queensland
Australia 4072
E.Proper@acm.org The Netherlands

PUBLISHED AS:

A.H.M. ter Hofstede, H.A. Proper, and Th.P. van der Weide. Computer Supported Query Formulation in an Evolving Context. In R. Sacks-Davis and J. Zobel, editors, *Proceedings of the Sixth Australasian Database Conference, ADC'95*, volume 17(2) of *Australian Computer Science Communications*, pages 188–202, Adelaide, Australia, January 1995.

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

[3]Part of this research was conducted while the author was working at the Computing Science Institute of the University of Nijmegen, and part of this work has been supported by an Australian Research Council grant, entitled: "An expert system for improving complex database design"

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.

Adequate information disclosure, be it in the context of evolving information systems or not, should not only provide language facilities on a conceptual level, but should also provide enough *expressive power* and be *suitable*. Expressive power is a purely theoretical matter. An expressiveness hierarchy for query languages exists (see e.g. [Cha88]). Suitability is a notion which is more difficult to capture formally. Suitability addresses the match between concepts offered by the language involved and concepts occurring in the application domain (Universe of Discourse) involved. A language can have sufficient expressive power for a certain application domain, but may not suitable; e.g. some queries might be very difficult to formulate elegantly. In this paper, focus is on the support of query formulation for conceptual query languages with a high expressive power, and which are suitable for *data intensive domains*. In data intensive domains, large quantities of (more or less) structured data, and few operations, have to be dealt with. Examples of data intensive domains can be found in office automation, CAD/CAM or hypermedia. In this paper the issue of automated support of query formulation is addressed in the context of evolving information systems. The approach outlined is also applicable to traditional information systems, as users of these systems may also have difficulties in maintaining an overview of the information stored.

The approach followed 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, BPW94, HPW94b]. This approach is based on an identical approach followed in information retrieval systems [BW92, Bru93]. 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. 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 NIAM ([NH89]), FORM ([Hal89, HO92]), and PSM ([HW93, HPW92, Hof93]). Even more, the results can readily be applied to ER or EER (see e.g. [Che76] and [HE92, EGH$^+$92]) based modelling techniques.

2 Evolving Information Systems

For evolving information systems [FOP92a, FOP92b, OPF94], 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).

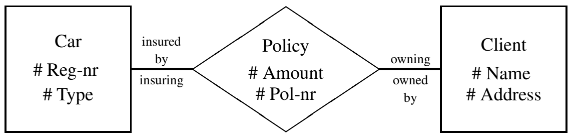


Figure 1: The information structure of a car insurance company

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.

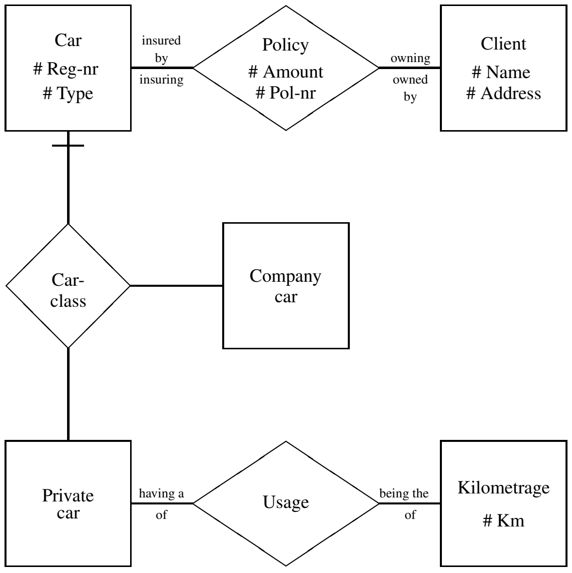


Figure 2: Car insurance with price differentiation

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.

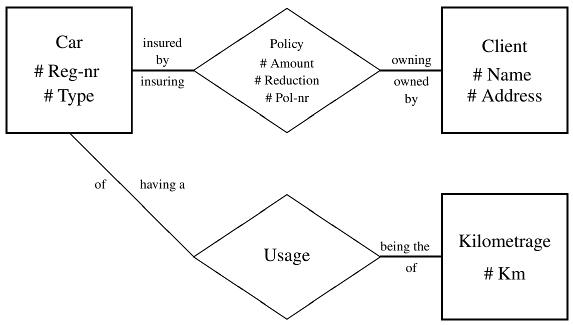


Figure 3: Car insurance with reduction rates

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.

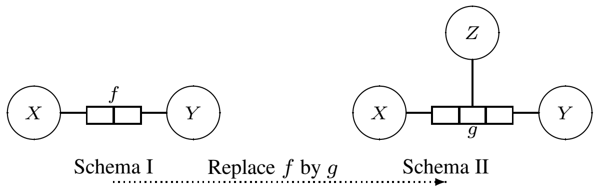


Figure 4: Evolution of schemata

Consider the two schemata in figure 4 as an abstract example of an extra-temporal schema. In this figure, the evolution of an application model history from binary fact type (relationship type) f to ternary fact type g is depicted. The associated extra-temporal information structure is depicted in figure 5.

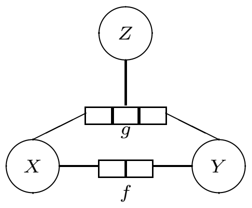


Figure 5: A simple extra-temporal schema

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 a sample query language, the language LISA-D ([HPW93, HPW94a, Hof93]) is employed. LISA-D can be seen as a general query language for object-role based data modelling techniques.

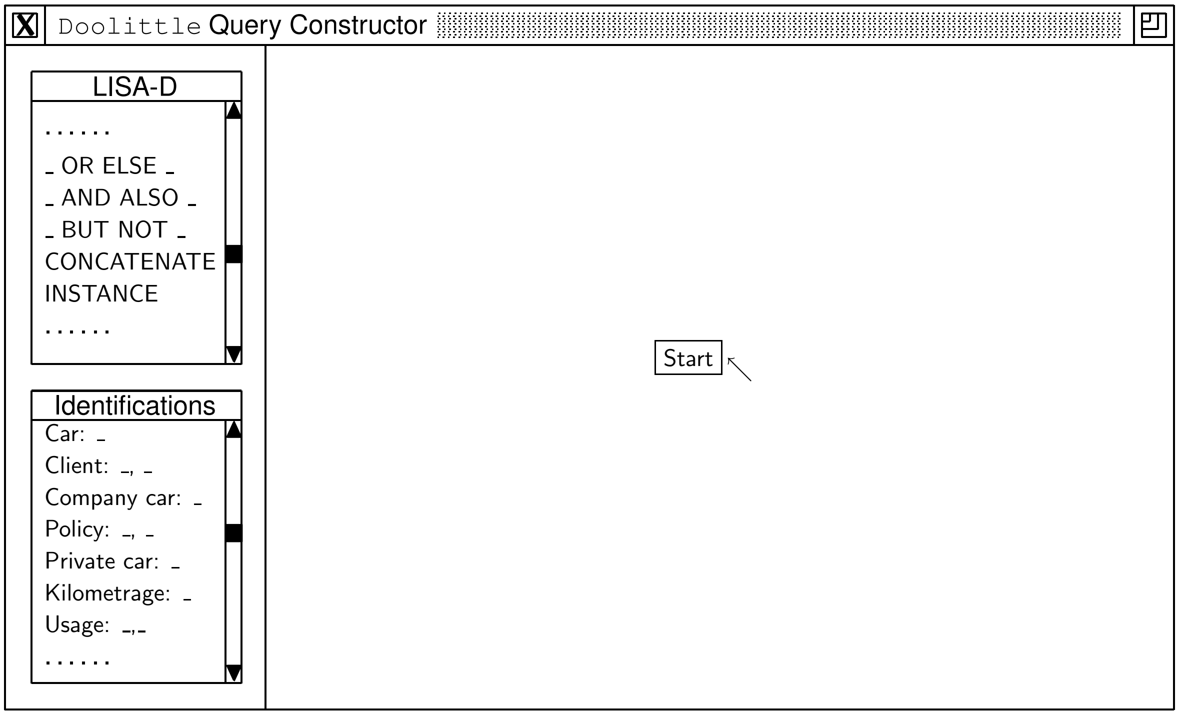


Figure 6: Opening screen of the query constructor

Suppose a searcher is interested in insured cars with a usage less than 1000 kilometers per year. In figure 6 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 7,which is referred to as the *starting node* of the navigator.

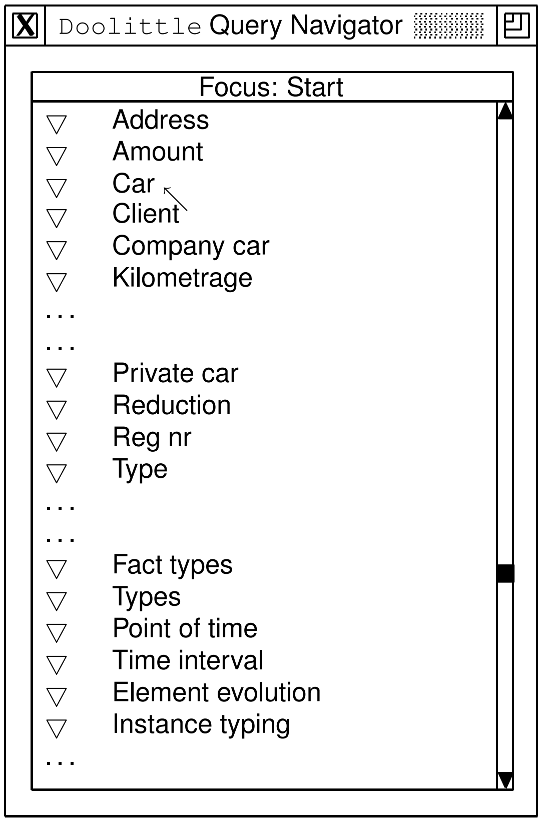


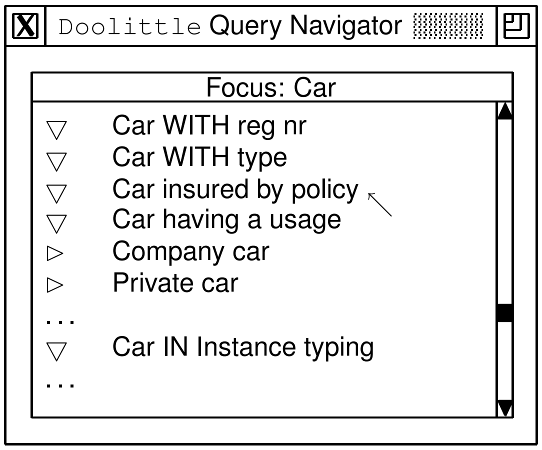
Figure 7: Selecting a start object type

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.

Since the searcher is mainly interested in cars, the item Car is selected. This choice leads to the screen of figure 8. 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 the screen provided in figure 9. After selecting item Car insured by policy owned by client, the searcher finishes this navigation session. In the mean time, the query builder screen has changed its focus accordingly, resulting in the screen depicted in figure 10 (page 8).

The searcher is not interested in all insured cars, but rather those with a usage of less than 1000 kilometers. The searcher therefore continues with selecting the AND ALSO operator from the Elisa-D menu, a binary operator on Elisa-D queries. The resulting screen of the query constructor is depicted in figure 11 (page 8).

The searcher can now start another navigation session to specify the interest in cars with a usage less than 1000km. This finally leads to the screen as provided in figure 12 (page 9).

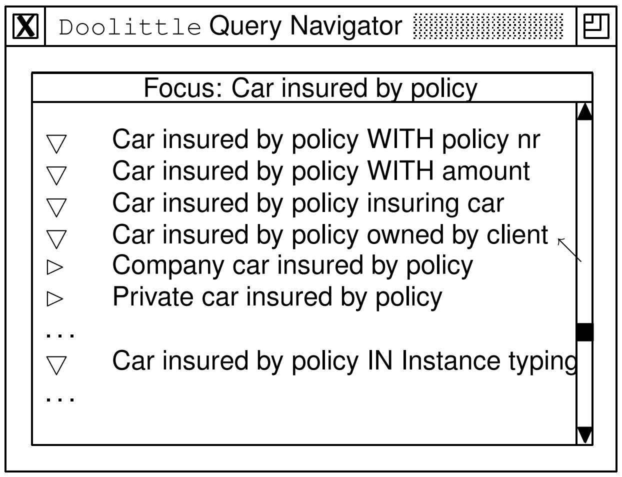


Doolittle Query Navigator

Focus: Car

▽ Car WITH reg nr
▽ Car WITH type
▽ Car insured by policy
▽ Car having a usage
▷ Company car
▷ Private car
...
▽ Car IN Instance typing
...

Figure 8: Starting point: cars



Doolittle Query Navigator

Focus: Car insured by policy

▽ Car insured by policy WITH policy nr
▽ Car insured by policy WITH amount
▽ Car insured by policy insuring car
▽ Car insured by policy owned by client
▷ Company car insured by policy
▷ Private car insured by policy
...
▽ Car insured by policy IN Instance typing
...

Figure 9: First refinement: policies

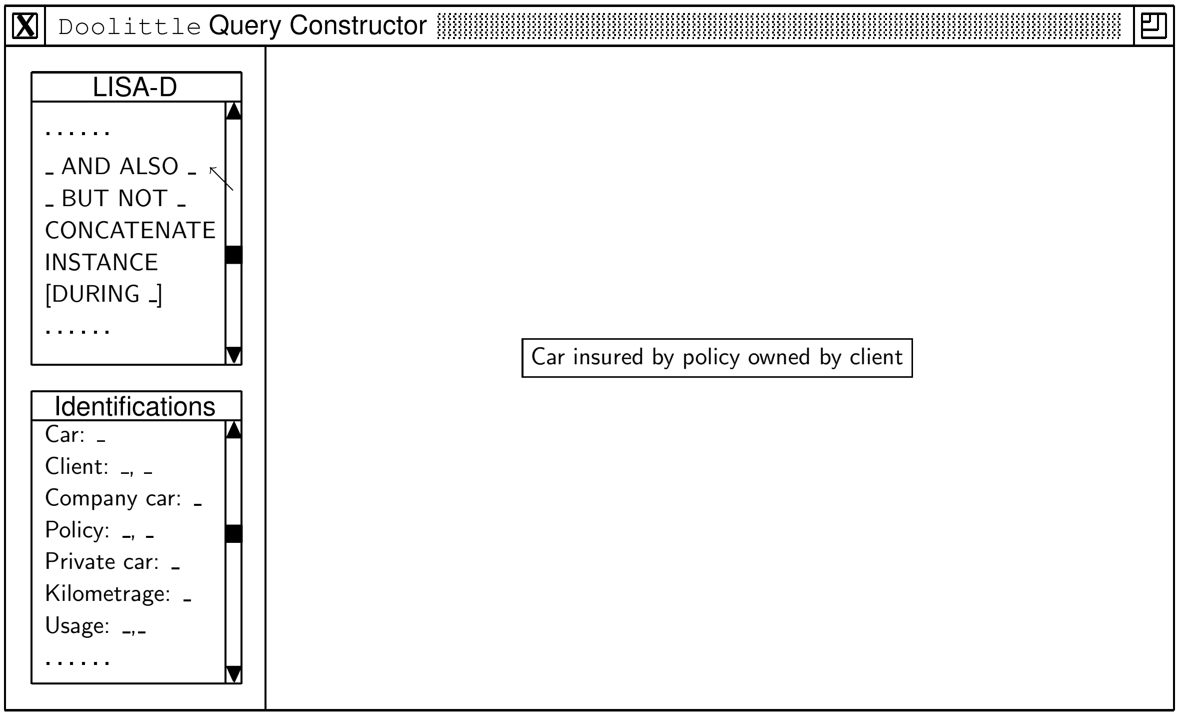


Figure 10: Query constructor after navigation session

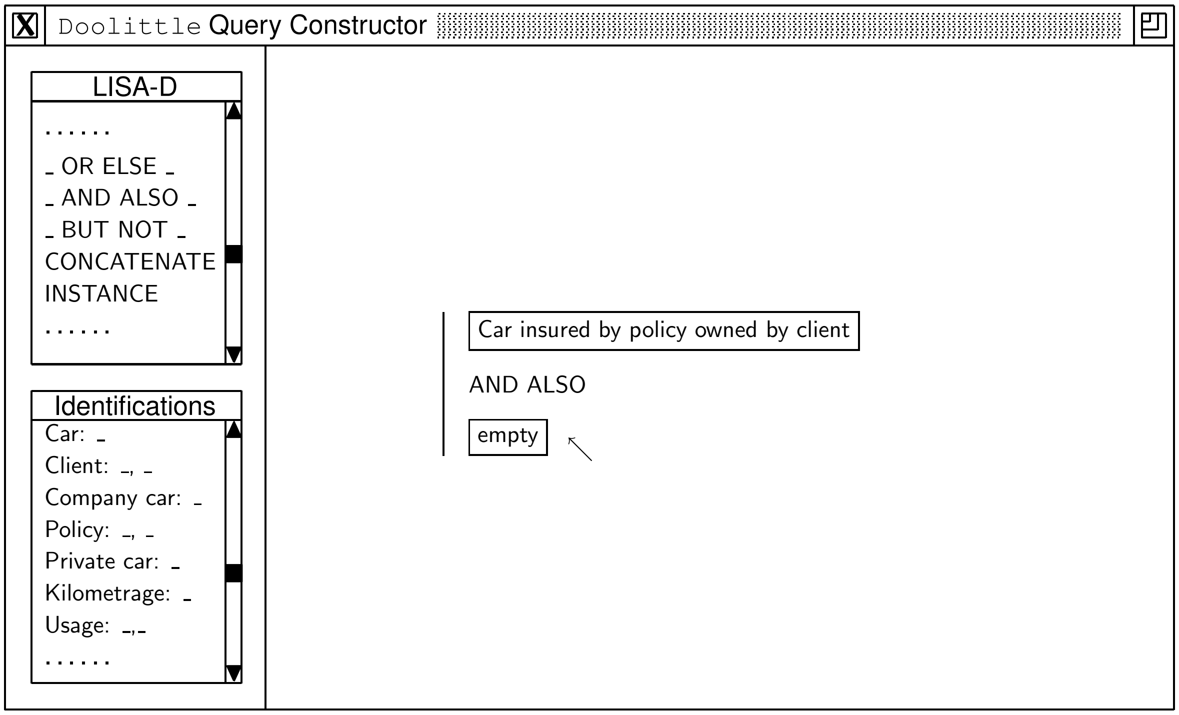
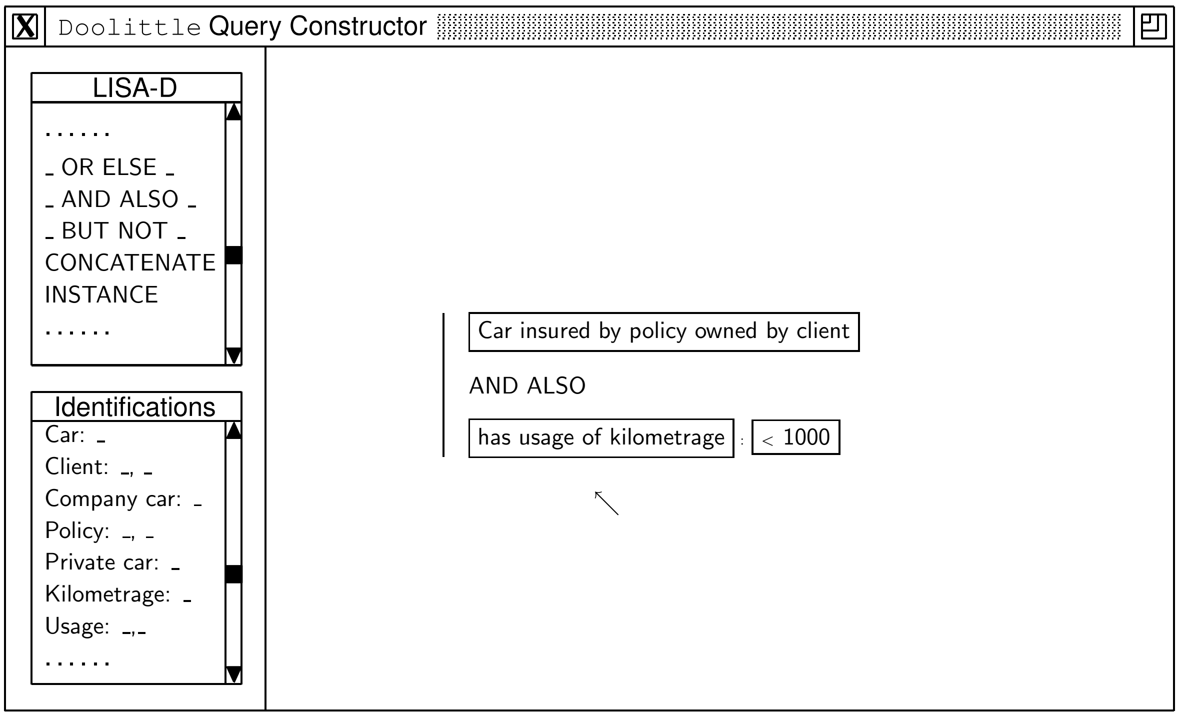


Figure 11: Adding an extra condition



Car insured by policy owned by client

AND ALSO

has usage of kilometrage : < 1000

Figure 12: Completed query

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 13 (page 10).

Finally, also queries relating meta and application level may be formulated by means of the query constructor. In figure 14 (page 10) such a query is depicted. Informally, this query yields all fact types that have ever been associated with the object type Car.

4 The Underlying Architecture

In this section a more formal background of the query-by-navigation/query-by-construction system is provided. We start by introducing the underlying stratified hypermedia architecture. 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 Stratified hypermedia architecture

A stratified hypermedia architecture, in its simplest form, is a two level hypermedia architecture as introduced in [BW92]. This architecture features a descriptive layer (hyperindex), indexing a lower layer

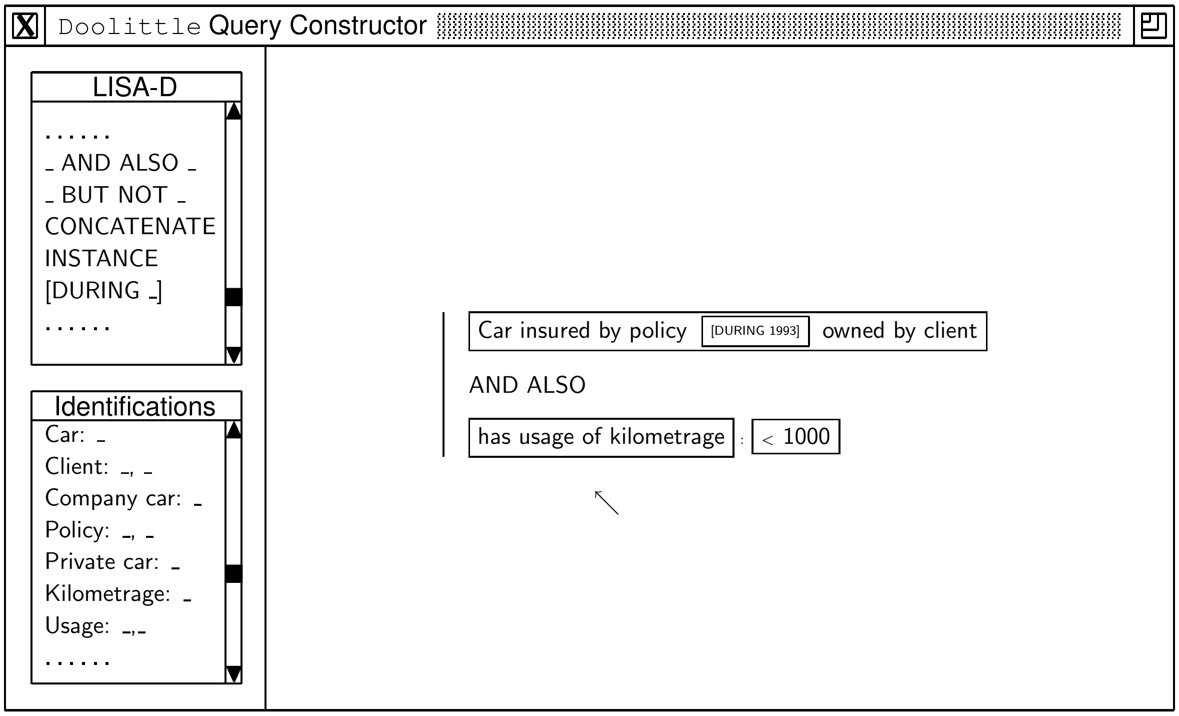


Figure 13: Query extended with a temporal selection

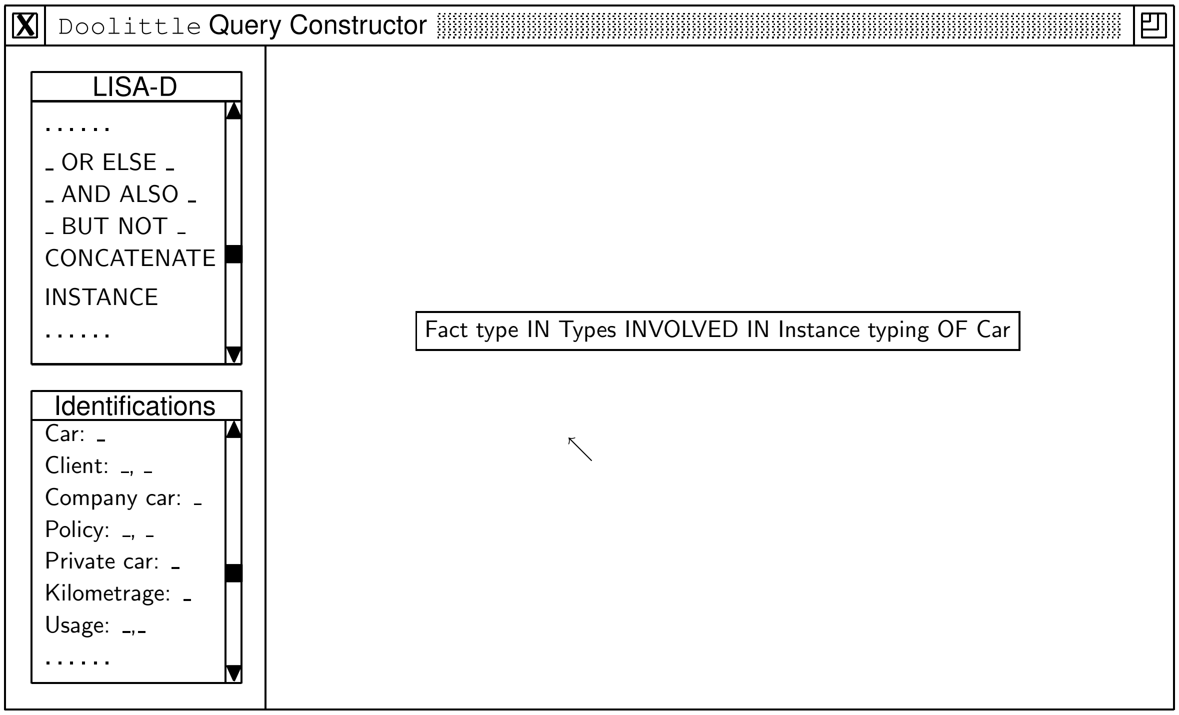


Figure 14: Relating meta level to application level

(the hyperbase). The hyperbase contains the actual information, whereas the hyperindex only provides an outline of the stored information. Both layers have a comparable structure.

The motivation for the introduction of a descriptive layer is that it offers the searcher the opportunity to search the index rather than the hyperbase itself. Searching an index can be performed much faster than directly searching the hyperbase itself. On the other hand, describing objects by descriptive terms is inherently imprecise and incomplete, thus introducing noise and omission.

Typical for hypertext systems is the possibility to navigate over the contents. By organizing the indexing information in a hypertext format, this navigation mechanism is also available for the index search process. By exploiting the structure of descriptors, it is possible to support the searcher during the index search process by providing context sensitive guidance.

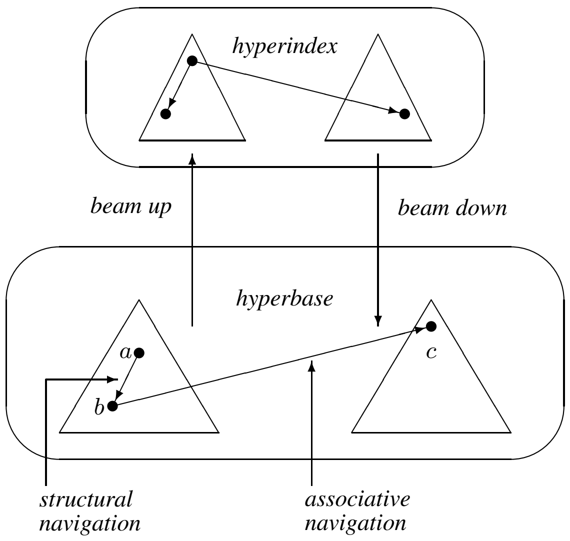


Figure 15: Operations within a hypermedia architecture

Formally, a layer in a stratified hypermedia architecture consists of the following components:

1. A set $\mathbb{F}$ of information fragments, called the *fragment base*. Fragments are the elementary parts of the stored information, which cannot be decomposed structurally into smaller components.
2. A set $\mathbb{N}$ of presentation units (or nodes), referred to as the *node base*. Nodes are constructed from information fragments. Nodes are used to present structural elements to the user.
3. A grammar $\mathbb{G}$ describes how the information in the layer is structured. Usually, this grammar is provided as a context-free grammar. Formally, $\mathbb{G}$ is a structure $\langle \mathbb{E}, \mathbb{P} \rangle$, where $\mathbb{E}$ is a set of syntactic categories, and $\mathbb{P}$ is a set of context-free production rules. $\mathbb{G}$ is referred to as the *schema* of the layer.
4. $\mathbb{V}$ is a set of *views*, called the *mask*. Views correspond to an external view of the information contents of a hypertext system.

A view, in its turn, is composed of the following components:

1. $\mathbb{S} \in \mathbb{E}$ is the *start symbol*, the syntactic category to which all objects within this view belong.
2. ω is the actual information contents of the view, being a set of (possibly overlapping) parse trees, generated from syntactic category $\mathbb{S}$ using the rules from grammar $\mathbb{G}$. ω is referred to as the *actual structure*.

3. $\mathbb{M}$ is the set of vertices within the parse trees of ω. A vertex is also called a *molecule*. The edges between molecules are referred to as *structural links*.
4. A visualization mechanism π for molecules, mapping molecules onto nodes: $\pi : \mathbb{M} \to \mathbb{N}$.
5. $\mathbb{L}$ is a set of *associative link schemata*, also denoted as cross-reference schemata. A particular link scheme $l \in \mathbb{L}$ consists of a set of links of the same category. Each link originates from a fragment in a node, and leads to a fragment within another node. Associative links are used to feature cross-references between a fragment in one molecule, and a fragment in another molecule.

Browsing the information contents of a hypertext system corresponds, in this approach, to a navigation process over molecules. There are two types of connections between molecules: structural and associative links. As a consequence, two types of navigation can be distinguished. The movement from one molecule to another (for example, from molecule a to molecule b in figure 15), using the underlying structure, is called *structural navigation*. Selecting an associative link (for example, from molecule b tot molecule c in figure 15) initiates the traversal of such a link. This is called *associative navigation*.

4.2 The Hyperindex

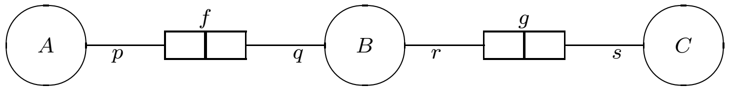


Figure 16: A small schema with identified predicators

Object role models (such as NIAM, ER, EER, PSM, FORM), 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 figure 16, 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, HPW94a, PW95, Hof93].

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. Figure 17 displays a molecule c and its immediate context.

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 ($a_1, \ldots, a_n$). This leads to the node design from figure 18. The verbalization function ρ will be introduced at the end of this section.

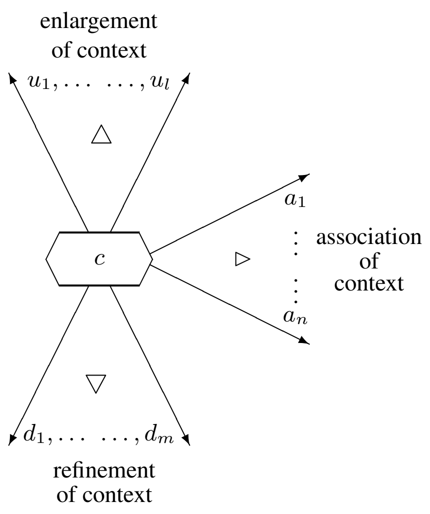


Figure 17: The environment of molecule c

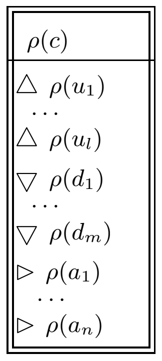

$\rho(c)$
$\triangle \; \rho(u_1)$
$\cdots$
$\triangle \; \rho(u_l)$
$\bigtriangledown \; \rho(d_1)$
$\cdots$
$\bigtriangledown \; \rho(d_m)$
$\triangleright \; \rho(a_1)$
$\cdots$
$\triangleright \; \rho(a_n)$

Figure 18: The presentation of molecule c

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:

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

Views

The hyperindex for an EVORM 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 8 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. The operator Rev is introduced as:

$$\begin{array}{rcl}
\mathsf{Rev}(x) & \to & x \\
\mathsf{Rev}(p) & \to & p^{\leftarrow} \\
\mathsf{Rev}(\langle P \rangle \circ p \circ \mathsf{Fact}(p)) & \to & \mathsf{Fact}(p) \circ p^{\leftarrow} \circ \mathsf{Rev}(\langle P \rangle) \\
\mathsf{Rev}(\langle P \rangle \circ p^{\leftarrow} \circ \mathsf{Base}(p)) & \to & \mathsf{Base}(p) \circ p \circ \mathsf{Rev}(\langle P \rangle)
\end{array}$$

An example of a reversal is:

$$\mathsf{Rev}(x \circ p \circ f \circ q^{\leftarrow} \circ y) = y \circ q \circ f \circ p^{\leftarrow} \circ x$$

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.

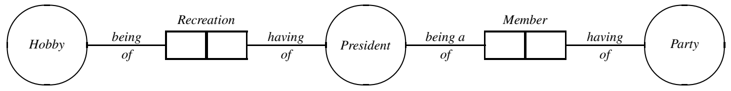


Figure 19: A naming

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 16 is provided in figure 19:

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

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*)

$$\begin{aligned}&\rho(P_1, n_1, \alpha_1) \wedge \rho(P_2, n_2, \alpha_2)\\ &\quad\vdash\ \rho(P_1 \circ P_2, n_1\ n_2, \alpha_1 + \alpha_2 + 1)\end{aligned}$$

This leads, for the schema in figure 16 with namings as provided by figure 19, to the following verbalizations:

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

The verbalization rules using predicator names and reverse predicator names, sometimes lead to clumsy sentences. Therefore, extra names for transitions through 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 16 we introduce the following connector names:

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

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:

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

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.

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

In our example, we have:

$$\mathsf{Cost}(A \circ p \circ f \circ q^{\leftarrow} \circ C) = 2$$

leading to the following verbalization:

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

4.3 The Hyperbase

The hyperbase contains the actual information in a format which closely resembles the format of the hyperindex. This format is the extension of linear path expressions, as introduced in the previous section, with (structured) constants. Such expressions are referred to as *instantiated linear path expressions*. These path expressions are used to describe concrete paths in the form of 'documented' paths.

The schema for instantiated linear path expressions contains the following syntactic categories: a syntactic category $\langle x\rangle$ for each object type x, category $\langle P\rangle$ yields the path expressions, category $\langle C\rangle$ is the class of structured constants, and syntactic category $\langle Q\rangle$ represents a query result. For each abstract object type $x \in \mathcal{N}$, we have the following rule:

$$\langle x\rangle \;\rightarrow\; x \;\circ\; \langle C\rangle$$

Constants are verbalised using their identifications (standard names). The constants in the linear path expressions are used to get a grip on the instances in the extra-temporal population. We omit the rules for structured constants. For any predicator $p \in \mathcal{P}$ and abstract object type $x \in \mathcal{N}$, we have the following rules:

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

Finally, we have the rule:

$$\langle Q\rangle \;\rightarrow\; \langle P\rangle^{*}$$

In this paper, we do not further elaborate on the definition of the hyperbase. For a more elaborate definition, refer to: [BPW93] and [BPW94] [Pro94].

The presentation of instantiated path expressions can be done analogously to the presentation of linear path expressions in the hyperbase. However, some special derivation rules will be necessary. We omit these rules in this paper.

4.4 Inter Layer Navigation

Hyperbase and hyperindex are related by means of the beam up and beam down operations.

The beam down operator leads from a query in the hyperindex to a query result in the hyperbase, in the form of a molecule of syntactic category $\langle Q \rangle$. The query result has associated all instantiated linear path expressions that correspond to the query result. Each instantiated linear path expression corresponds to a molecule in the hyperbase. The searcher may then select the beam down destination from this list.

We first introduce for any extra-temporal population Pop of the disclosure schema, the predicate InPop over path expressions. The expression $\mathsf{InPop}(A, P)$ states that the instantiated path expression A is an instance of query P. Using this predicate, the beam down operator can be introduced. Let x be an object type, c a structured constant, p a predicator, and P, Q linear path expressions, then the beam down operator is recursively defined as:

$$\begin{aligned}
\mathsf{BD}(x) &\triangleq \left\{ x \circ c \;\middle|\; \mathsf{InPop}(x \circ c, x) \right\} \\
\mathsf{BD}(p) &\triangleq \{p\} \\
\mathsf{BD}(p^{\leftarrow}) &\triangleq \{p^{\leftarrow}\} \\
\mathsf{BD}(P \circ Q) &\triangleq \left\{ A \circ B \;\middle|\; \begin{array}{l} A \in \mathsf{BD}(P) \wedge \\ B \in \mathsf{BD}(Q) \wedge \\ \mathsf{InPop}(A \circ B, P \circ Q) \end{array} \right\}
\end{aligned}$$

When beaming up from a molecule in the hyperbase, a proper guide must be constructed. This means that all instances must be filtered from the instantiated path expression corresponding to the current focus in the hyperbase.

As an example, suppose the population of the schema from figure 19 contains the fact that president 'Coolidge C.' has hobby 'Golf'. Beaming down from the query Hobby being Recreation of President leads (among others) to the following instance:

Hobby 'Golf' being Recreation of President 'Coolidge C.'

Beaming up from this instance yields the original query.

5 Conclusions

In this paper, it has been argued that applications with evolving conceptual schemata, or large conceptual schemata, may benefit from guided support for query formulation. For this purpose, the integration between stratified hypermedia architecture and information systems was investigated. This integration enabled interactive query formulation.

Future research concentrates on automated support for composition of linear path expressions into more complex path expressions. Furthermore, the presented theory must be tested in practice by an implementation.

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.

[BPW94] C.A.J. Burgers, H.A. Proper, and Th.P. van der Weide. An Information System organized as Stratified Hypermedia. In N. Prakash, editor, *CISMOD94, International Conference on Information Systems and Management of Data*, pages 159–183, Madras, India, October 1994.

[Bru93] P.D. Bruza. *Stratified Information Disclosure: A Synthesis between Information Retrieval and Hypermedia*. PhD thesis, University of Nijmegen, Nijmegen, The Netherlands, EU, 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.

[Cha88] A.K. Chandra. Theory of Database Queries. In *Proceedings of the Seventh ACM Symposium on Principles of Database Systems*, pages 1–9, Austin, Texas, March 1988.

[Che76] P.P. Chen. The entity-relationship model: Towards a unified view of data. *ACM Transactions on Database Systems*, 1(1):9–36, March 1976.

[EGH$^+$92] G. Engels, M. Gogolla, U. Hohenstein, K. Hülsmann, P. Löhr-Richter, G. Saake, and H-D. Ehrich. Conceptual modelling of database applications using an extended ER model. *Data & Knowledge Engineering*, 9(4):157–204, 1992.

[FOP92a] E.D. Falkenberg, J.L.H. Oei, and H.A. Proper. A Conceptual Framework for Evolving Information Systems. In H.G. Sol and R.L. Crosslin, editors, *Dynamic Modelling of Information Systems II*, pages 353–375. North-Holland, Amsterdam, The Netherlands, EU, 1992. ISBN 0444894055

[FOP92b] E.D. Falkenberg, J.L.H. Oei, and H.A. Proper. Evolving Information Systems: Beyond Temporal Information Systems. In A.M. Tjoa and I. Ramos, editors, *Proceedings of the Data Base and Expert System Applications Conference (DEXA'92)*, pages 282–287, Valencia, Spain, EU, September 1992. Springer Verlag, Berlin, Germany, EU. ISBN 3211824006

[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.

[Hal89] T.A. Halpin. *A logical analysis of information systems: static aspects of the data-oriented perspective*. PhD thesis, University of Queensland, Brisbane, Australia, 1989.

[HE92] U. Hohenstein and G. Engels. SQL/EER-syntax and semantics of an entity-relationship-based query Language. *Information Systems*, 17(3):209–242, 1992.

[HO92] T.A. Halpin and M.E. Orlowska. Fact-oriented modelling for data analysis. *Journal of Information Systems*, 2(2):97–119, April 1992.

[Hof93] A.H.M. ter Hofstede. *Information Modelling in Data Intensive Domains*. PhD thesis, University of Nijmegen, Nijmegen, The Netherlands, 1993.

[HPW92] A.H.M. ter Hofstede, H.A. Proper, and Th.P. van der Weide. Data Modelling in Complex Application Domains. In P. Loucopoulos, editor, *Proceedings of the Fourth International Conference CAiSE'92 on Advanced Information Systems Engineering*, volume 593 of *Lecture Notes in Computer Science*, pages 364–377, Manchester, United Kingdom, EU, May 1992. Springer Verlag, Berlin, Germany, EU. ISBN 3540554815

[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.

[HPW94a] A.H.M. ter Hofstede, H.A. Proper, and Th.P. van der Weide. A Conceptual Language for the Description and Manipulation of Complex Information Models. In G. Gupta, editor, *Seventeenth Annual Computer Science Conference*, volume 16 of *Australian Computer Science Communications*, pages 157–167, Christchurch, New Zealand, January 1994. University of Canterbury. ISBN 047302313

[HPW94b] 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

[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

[OPF94] J.L.H. Oei, H.A. Proper, and E.D. Falkenberg. Evolving Information Systems: Meeting the Ever-Changing Environment. *Information Systems Journal*, 4(3):213–233, 1994.

[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

[Pro94] H.A. Proper. *A Theory for Conceptual Modelling of Evolving Application Domains*. PhD thesis, University of Nijmegen, Nijmegen, The Netherlands, EU, 1994. ISBN 909006849X

[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.

[PW95] H.A. Proper and Th.P. van der Weide. Information Disclosure in Evolving Information Systems: Taking a shot at a moving target. *Data & Knowledge Engineering*, 15:135–168, 1995.