Query Formulation as an Information Retrieval Problem

Version of June 23, 2004 at 10:27

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

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

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

PUBLISHED AS:

A.H.M. ter Hofstede, H.A. Proper, and Th.P. van der Weide. Query formulation as an information retrieval problem. *The Computer Journal*, 39(4):255–274, September 1996.

Abstract

Query formulation in the context of large conceptual schemata is known to be a hard problem. When formulating ad-hoc queries users may become overwhelmed by the vast amount of information that is stored in the information system; leading to a feeling of *lost in conceptual space*. In this article we develop a strategy to cope with this problem. This strategy is based on ideas from the information retrieval world. In particular the query by navigation mechanism and the stratified hypermedia architecture. The stratified hypermedia architecture is used to describe the information contained in the information system on multiple levels of abstraction.

When using our approach to the formulation of queries, a user will first formulate a number of simple queries corresponding to linear paths through the information structure. The formulation of the linear paths is the result of the *explorative phase* of query formulation. Once users have specified a number of these linear paths, they may combine them to form more complex queries. This last process is referred to as *query by construction*, and corresponds to the *constructive phase* of the query formulation process.

1 Introduction

Most present day organisations make use of some automated information system. This usually means that a large body of vital corporate information is stored in these information systems. As a result an obvious, yet crucial, function of information systems is the support of disclosure of this information. Without a set of adequate information disclosure avenues an information system becomes worthless since there is no use in storing information that will never be retrieved.

Adequate support for information disclosure, however, is far from a trivial problem. Most information systems do not provide any support for the users in their quest for information. Furthermore, the conceptual schemata of real-life applications tend to be quite large and complicated. As a result, the users may easily become 'lost in conceptual space' and they can end up retrieving irrelevant (or even wrong) objects and may miss out on relevant objects. Retrieving irrelevant objects leads to a low *precision*, missing relevant objects has a negative impact on the *recall* ([SM83]).

[3]Part of this work has been supported by an Australian Research Council grant, entitled: "An expert system for improving complex database design"

The disclosure of information stored in an information system has some clear parallels to the disclosure problems encountered in document retrieval systems. To draw this parallel in more detail, we quote the information retrieval paradigm as introduced in e.g. [Rij75] and [SM83]. The paradigm starts with an individual or company having an information need they wish to fulfill. This need is typically a vague notion and needs to be made more concrete in terms of an information request (the query) in some (formal) language. Formulation of this need leads to an *information request* q. The information request should be as good as possible a description of the information need. The information request is then passed on to an automated system, or a human intermediary, who will then try to fulfill the information request using the information stored in the system. To this end, the information request is matched against the *characterisation* of *information objects* which are available in the information base (also referred to as information carriers or documents). This is illustrated in the information disclosure, or information retrieval paradigm, presented in figure 1 which is taken from [BW92].

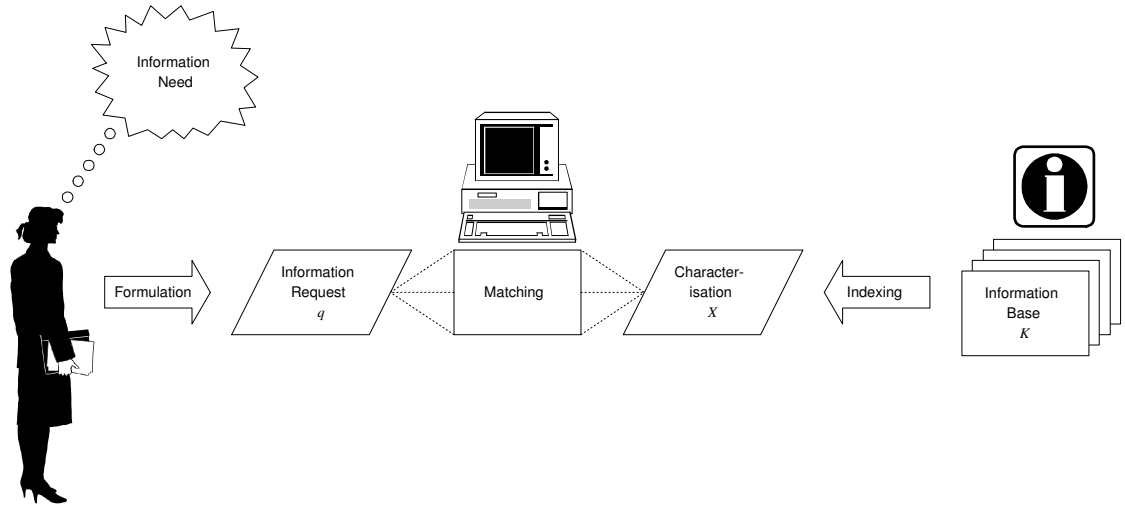

Figure 1: The information retrieval paradigm

The information retrieval paradigm for document retrieval systems is, in our opinion, directly applicable to traditional information systems. In the paradigm, the retrievable information is modelled as a set $\mathcal{K}$ of information objects, which together constitute the information base (or population). In a document retrieval system the information base will be a set of documents ([SM83], [Rij75]), while in the case of an information system the information base will contain a set of facts conforming to a conceptual schema. Each information object $o \in \mathcal{K}$ is characterised by a set of descriptors $\chi(o)$ that facilitates its disclosure. The characterisation of information objects is carried out by a process referred to as indexing. In an information system, the stored objects (the population or information base) can always be identified by a set of (denotable) values; the identification of those objects. For example, an address may be identified as a city name, street name, and house number. The characterisation of objects in an information system is directly provided by the reference schemes of the object types.

The actual information disclosure is driven by a process of matching. In document retrieval applications this matching process tends to be rather complex. Furthermore, the characterisation of documents is known to be a hard problem ([Mar77], [Cra78], [SM83]), although newly developed approaches turn out to be quite successful ([Sal89]). In information systems the matching process is less complex as the objects in the information base have a more clear characterisation (the identification). In this case, the identification of the objects (facts) is simply related to the query formulation q by some (formal) query language.

The remaining problem is then the query formulation process itself. An easy and intuitive way to formulate queries is absolutely essential for adequate information disclosure. Quite often, the quest from users to fulfill their information need can be aptly described by ([Bru93]):

I don't know what I'm looking for, but I'll know when I find it.

In document retrieval systems this problem is attacked by using query by navigation ([GGP89], [Luc90], [CD90], [ACG91], [BW92]) and relevance feedback mechanisms ([Rij89]). The query by navigation interaction mechanism between a searcher and the system is well-known from the Information Retrieval field, and has proven to be useful. The above discussed parallel between information disclosure in the context of information systems, as well as information retrieval systems, leads to the natural conclusion that these mechanisms also apply to the query formulation problem for information systems.

In line with the above discussed information retrieval paradigm and the notion of relevance feedback, a query formulation process (both for a document retrieval system, and an information system) can be said to roughly consist of the following four phases:

1. The *explorative phase*. What information is there, how is it related, and what does it mean?
2. The *constructive phase*. Using the results of phase 1, the actual query is formulated.
3. The *feedback phase*. The result from the query formulated in phase 2 may not be completely satisfactory. In this case, phases 1 and 2 need to be re-done and the result refined.
4. The *presentation phase*. In most cases, the result of a query needs to be incorporated into a report or some other document. This means that the results must be grouped or aggregated in some form.

Depending on the user's knowledge of the system, the importance of the respective phases may change. For instance, a user who has a good working knowledge of the structure of the stored information may not require an elaborate first phase and would like to proceed with the second phase as soon as possible.

One important step that has already led to an improvement of information disclosure in information systems, has been the introduction of query languages on a conceptual level. Examples of such conceptual query languages are RIDL ([Mee82]), LISA-D ([HPW93], [PW95]), and FORML ([Hal95]). Thus far, these languages are mainly used in the context of the ORM ([Win90], [HM94], [Hal95]) approach to information modelling. A key difference between ORM approaches and ER based approaches are the close ties to natural language. In the design procedures for ORM models, one starts out by modelling a domain in terms of a set of natural language sentences. These natural language sentences verbalise this domain in terms used by the *domain experts*; i.e. the people that will be using the information system. Essentially, these *verbalisations* provide the basic expressions for the conceptual query languages. Similar attempts for ER models are usually hampered by the lack of rich verbalisations in such models.

By letting users formulate queries on a conceptual level, users are safeguarded from having to know the exact mapping to underlying relational tables to be able to formulate queries in a non conceptual language like SQL. This allows users to concentrate on the actual query formulation, during the constructive phase, rather than on such mapping details. The next step on this path is to introduce ways to support users in the formulation of queries in such conceptual query languages (CQL); more exploiting the rich semantics provided by the conceptual schema of an application. In this paper, we show how the application of query by navigation may lead to an elegant query formulation process. Query by navigation is particularly useful for the explorative and feedback phases of the query formulation process. We also introduce a *query by construction mechanism* allowing for the formulation of the final query using computer support.

The structure of the paper is as follows. In section 2 we describe a general architecture for hypermedia systems used for query by navigation systems, and discuss how a conceptual schema and its population fit in this architecture. A larger example will provide a preview of the capabilities of a query by navigation mechanism. In section 3 and section 4 the formal definition of the query by navigation system, in relation to a conceptual schema, is given. The migration between abstraction levels, and the underlying prerequisites, are discussed in section 5. Finally, before concluding, section 6 discusses the query by construction mechanism complementing the query by navigation mechanism with syntactically richer expressions.

2 An Architecture for Information System Exploration

Stratified hypermedia is an architecture in which information is organised via several layers of abstraction, allowing access to the information via each of these layers. Base layers contain the actual information, while the other layers provide descriptions (abstractions) of this information with the purpose to simplify access to those base layers, and to provide insight in specific characteristics of this information.

In this section we briefly explain this architecture and discuss an example of how this architecture can be applied in the context of query formulation for information systems. The remainder of this article then develops this idea in more exact and formal terms.

2.1 Stratified Hypermedia Architecture

Stratified hypermedia architecture, in its simplest form, is a *two level hypermedia architecture* as introduced in e.g. [Luc90], [GGP89], [AAC+89], [ACG91], [BW92], [RWB93]. The two level architecture usually features a descriptive layer (the hyperindex) indexing the lower layer (the hyperbase). The hyperbase contains the actual information, whereas the hyperindex only provides an outline (characterisation) of the stored information. A stratified hypermedia architecture supporting multiple layers of abstraction is discussed in e.g. [SDBW91].

Users of a hypertext application based on the stratified hypermedia architecture may be compared to explorers. When navigating within one layer of abstraction, users are completely free to follow any link between pieces of information, thus allowing them to connect relevant information in their own (subjective) way. By navigating within and between the layers, users slowly but surely increase their knowledge about relevant parts of the information stored in the system at their own accord and preferred level of abstraction. In doing so, at a proper level of abstraction, they become cognitively better equipped to descend to the lower abstraction levels and select the desired pieces of information. The stratified hypermedia architecture, and its accompanying query by navigation system, have proven to be useful in practical situations (see e.g. [Pol93], [BBB91]). This creates the expectation that a similar query formulation strategy would also work well in the context of traditional information systems. Although there is currently no implementation available of the ideas presented here, there is considerable commercial interest. For example, the producers of the InfoModeler CASE-Tool ([Asy94]), are developing a query formulation tool which will already incorporate some of the underlying ideas.

As an example of a hyperindex, consider figure 2. This example is taken from [BBB91], which describes a prototype implementation of a query by navigation based retrieval system supporting that is still being used by History of Art libraries, and is now sold as a commercial application. This is a simple example hyperindex, only dealing with a breakdown of the index expression

proclamation of resurrection of Jesus by disciples

In reality, a hyperindex is formed by the union of a large number of such smaller lattices, which then form a so-called *lithoid*. A sample navigation session is provided in figure 3. A user starts at the starting node, which contains a list of all elementary terms from the hyperindex. The user can then select one of these words as a first refinement. Once a more complicated index expression has been selected, e.g. resurrection of Jesus, it becomes possible to select the more elementary expressions that are part of the currently focussed expression. In the case of resurrection of Jesus this would be resurrection and Jesus. In such a navigation session, the user basically traverses edges in the graph of the hyperindex as shown in figure 2.

Each entry in the nodes displayed in figure 3 represents one way to continue the search through the hyperindex. A node thus corresponds to a moment of choice in the search process. The order in which the alternatives are listed in the starting node, and nodes in general, can be based on multiple factors. An example of such a factor is the user's past search behaviour ([BHW96]). In this article we also briefly discuss another factor; which is based on the conceptual relevance of object types, and other components, within a conceptual schema.

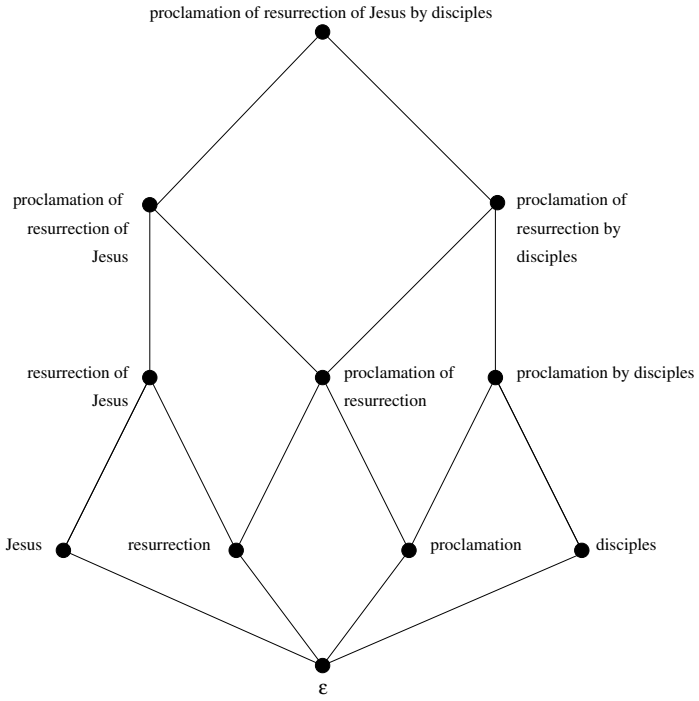

Figure 2: An example part of a hyperindex

In stratified hypermedia architecture, layers are related to each other via so-called characterisations. For instance, chunks of information from the hyperbase may be characterised by keywords. In that case, similarly to the given example, the hyperindex is organised around the concept of keyword. This characterisation relation forms the basis for *inter layer navigation*. Typically, a user formulates an information request by navigating through the hyperindex to an information descriptor aptly describing the user's information need. Then the user transfers to the hyperbase, focusing on the information objects matching this description. In terms of the example given above, the user navigates in a query by navigation session from the start node to the node resurrection of Jesus. A transfer to the hyperbase would now lead to the presentation of a set of documents regarding the resurrection of Jesus; or at least the documents that have a characterisation that matches resurrection of Jesus. The reverse inter layer navigation (hyperbase to hyperindex) transfers the user to the description of the current information object.

Transferring to the hyperindex after arriving at an interesting node in the hyperbase corresponds to a search strategy where the user first searches an information object which is felt to be a typical example of to the information need, and then asks for all similar information objects (query by example). This latter process involves a transfer from the object in the hyperbase that represents the information need, to an object in the hyperindex that characterises this hyperbase object. This transfer from hyperbase to hyperindex is immediately followed by a transfer back to the hyperbase resulting in all objects in the hyperbase which are relevant to the current characterisation of the hyperindex object.

Switching between layers can also be used as a feedback mechanism as used in information retrieval ([Rij86], [Rij90]). When, after navigating through the hyperindex, the user finds that the current focus is possibly a proper description of the information need, they can test this by asking the system to present (part of) the relevant objects in the hyperbase. This way the user will, rather than completely transferring from the hyperindex to the hyperbase, get a first impression of what has been achieved so-far. If the user is not satisfied, further refinements can be made based on the feedback provided by the system.

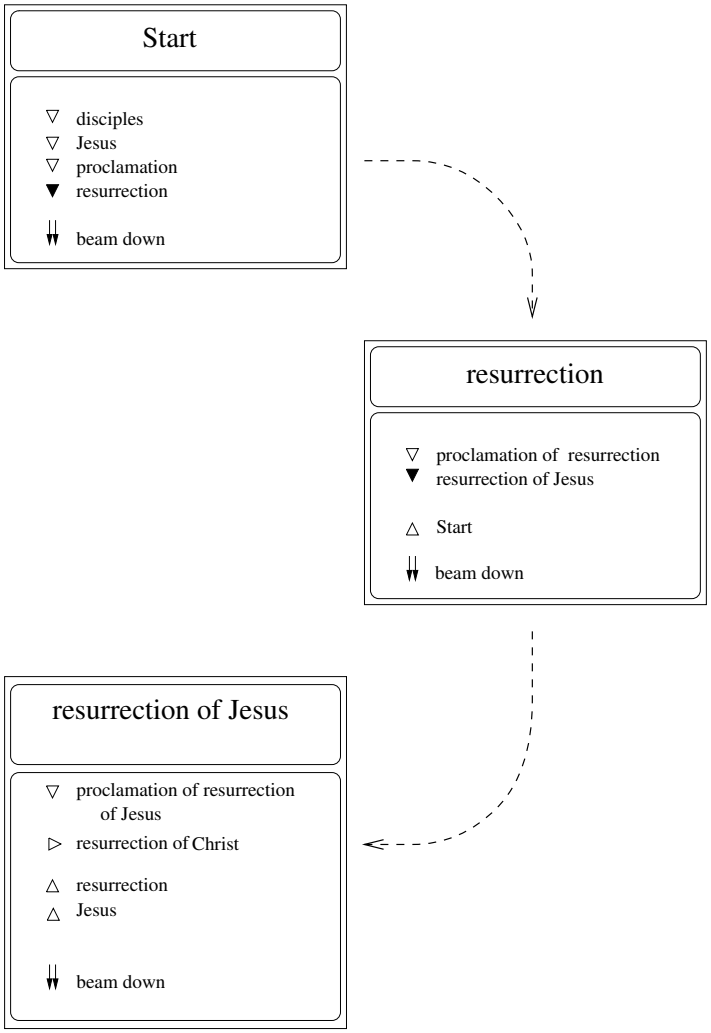

Figure 3: Example navigation session

2.2 A Formalisation of Stratified Hypermedia Architecture

This brings us to the formal introduction of stratified hypermedia architecture as a network of layers that are related by characterisation relations. A layer is introduced as a structure $\langle \mathbb{F}, \mathbb{N}, \mathbb{G}, \mathbb{V} \rangle$ where:

1. $\mathbb{F}$ is the set of (information) fragments, called the *fragment base*. Fragments are elementary parts of the stored information which can (will) not be decomposed structurally into smaller components.

   In the example given above, $\mathbb{F}$ would at least contain:

   > Jesus, resurrection, ..., proclamation of resurrection, ...,
   > proclamation of resurrection of Jesus by disciples

2. $\mathbb{N}$ is the set of presentations (or nodes), referred to as the *node base*. Nodes are units of presentation and are used to present structural elements to the user. They are constructed from information fragment references, and appear in some ordered fashion within this presentation. Formally, a presentation is a structure $\langle F, \rho \rangle$, where F is a set of information fragments, and ρ a partial order over F.

   In figure 3 three example nodes are shown.

3. $\mathbb{G}$ is a structure $\langle \mathbb{E}, \mathbb{P} \rangle$, where $\mathbb{E}$ is a set of syntactic categories, and $\mathbb{P}$ a set of production rules. $\mathbb{G}$ is referred to as the *schema* of the layer. The grammar $\mathbb{G}$ is used to structure the information in a layer. Usually, this grammar is provided as a context-free grammar (SGML, ODA, HTML).

   In our example, the grammar we used is the one for index expressions ([Cra78], [Far80], [Bru90]).

4. $\mathbb{V}$ is a set of views, called the *mask*. This is explained below.

Documents may be considered from different points of view, where each view features its own structure of the document. For example, a document may be considered as a (long) piece of text, or may be viewed as composed of sections, subsections, etc. It is the user who decides what view best suits the intended use of the document. As a result, within a view, each abstract information element belongs to some syntactic category. A set of composition rules describes the composition of syntactic categories in terms of other syntactic categories. Elementary syntactic categories have associated an information fragment, i.e. an elementary chunk of information. A view V, in its turn, is a structure $\langle \mathbb{S}, \mathbb{M}, \omega, \pi, \mathbb{L} \rangle$ where:

1. $\mathbb{S} \in \mathbb{E}$ is the overall syntactic category for documents within this view (the *start symbol* of the subgrammar).

   In the example case, this is the start symbol of the index expression grammar.

2. $\mathbb{M}$ is the set of instantiations of syntactic categories that are available within this view. Such an instantiation is called a *molecule*.

   The nodes in the hyperindex graph shown in figure 2 are examples of molecules; not to be confused with the presentation nodes as depicted in figure 3.

3. ω is a binary relation over $\mathbb{M}$ that describes how syntactic categories are composed from each other, respecting the rules from $\mathbb{G}$. ω is also referred to as the actual structure of the view.

   In the example, ω corresponds to the set of edges in figure 2.

4. $\pi : \mathbb{M} \rightarrow \mathbb{N}$ maps each molecule from $\mathbb{M}$ to a presentation unit.

   The node from the hyper index graph in figure 2 labelled resurrection is presented by the node with the same label. In the presentation node, the direct environment of the resurrection node is displayed as well, and the user is offered the possibilities to travel downwards or upwards in the hyper index.

5. $\mathbb{L}$ is a set of *associative link schemata*. Links are used to describe cross reference relations between documents. A particular link scheme consists of a set of links of the same category. For example, an isa-relation (link schema) might express the categorial classification of index terms, while a co-relation is a symmetric relation, expressing that an index term corresponds to another index term.

   In the example, the presentation node labelled resurrection of Jesus contains an associative link to a node called resurrection of Christ.

Layers are related via characterisation relations. A characterisation relates layers via specific views within these layers, and describes how molecules of one view are associated with molecules of the other view. So, if χ is a characterisation of layer L_1 via view V_1 into view V_2 of layer L_2, then:

$$\chi \subseteq V_1.\mathbb{M} \times V_2.\mathbb{M}$$

Note that more characterisation relations may exist between two layers, for example, originating from different views within these layers.

Two kinds of navigations between molecules are presented in figure 4, two parse trees of the hyperbase layer are presented (B_1, B_2), and two parse trees of the hyperindex layer (I_1, I_2). The movement from one molecule to another, using the underlying structure of a parse tree, is called *structural navigation*. Selecting an associative link initiates traversal of such a link, called *associative navigation*, leads to a change in context (parse tree). Associative links are used to feature cross-references between a fragment in one molecule, and a fragment in another molecule. The beam up and beam down operations are used to facilitate inter layer navigation.

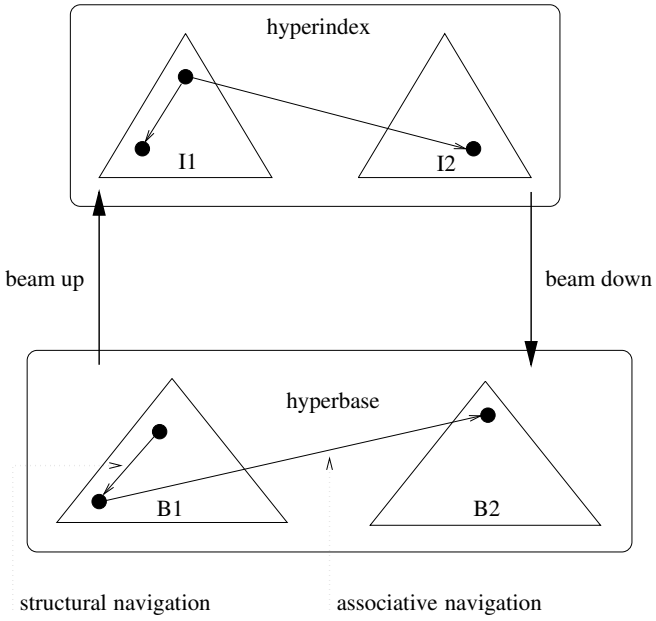

Figure 4: Operations of the hypermedia

2.3 Exploring an Information Structure

Now we have discussed the stratified hypermedia architecture both formally and informally, we can more concretely discuss the ideas presented in this article. To this end an example query formulation process is discussed that utilises the stratified hypermedia architecture. The remainder of this article is concerned with a proper formalisation of these ideas.

In our view, the process of query formulation corresponds to a search through the information system to gradually fulfill some information need. Using query by navigation, the arguments are constructed that are to be integrated by the constructive phase of query formulation. During query by navigation, a (partial) query is formulated by stepwise refining or enlarging the current formulation (the *focus*), until the searcher is satisfied with the current formulation. In the example we make use of the conceptual schema of a database for presidents of the United States of America as depicted in figure 5. This schema is provided as an Object-Role Modelling schema ([NH89], [Hal95]). It describes a domain in terms of relationship types (also referred to as fact types) and object types. Roles (the rectangular boxes) indicate how object types participate in relationship types. Some examples of relationship types are: ... has vice president ..., ... has nr of ... in ..., and examples of object types are: Administration, Person. On some types a subtyping relationship is present. President is subtype of Politician, which in its turn is a subtype of Person. Besides these two classes of types we see constraints, like the arrow headed lines, the black dots, and the encircled U. These will be explained in more detail when needed.

The first node shown to the user is depicted in figure 6, which simply lists all object types in the conceptual schema. Let us presume the user is interested in presidents who are married to someone, and the number of children that resulted from these marriages. In the starting node, the user may select 'president' as the first refinement of the information need. This leads to the example node as presented in figure 7. The associated node shows the direct environment of object type President. The set of possible refinements of the current focus is built as follows. For each n-ary relationship type in which the current focus (president) plays a role, we have $n-1$ possible refinements since there are $n-1$ possible ways to continue the path *through* this relationship type. The associative links are now derived from the subtyping hierarchy in the conceptual schema. In our example ORM schema these are the supertypes of the object type President, being Politician and Person.

Let us presume the searcher selects the president who is involved in a marriage as the next focus. This action

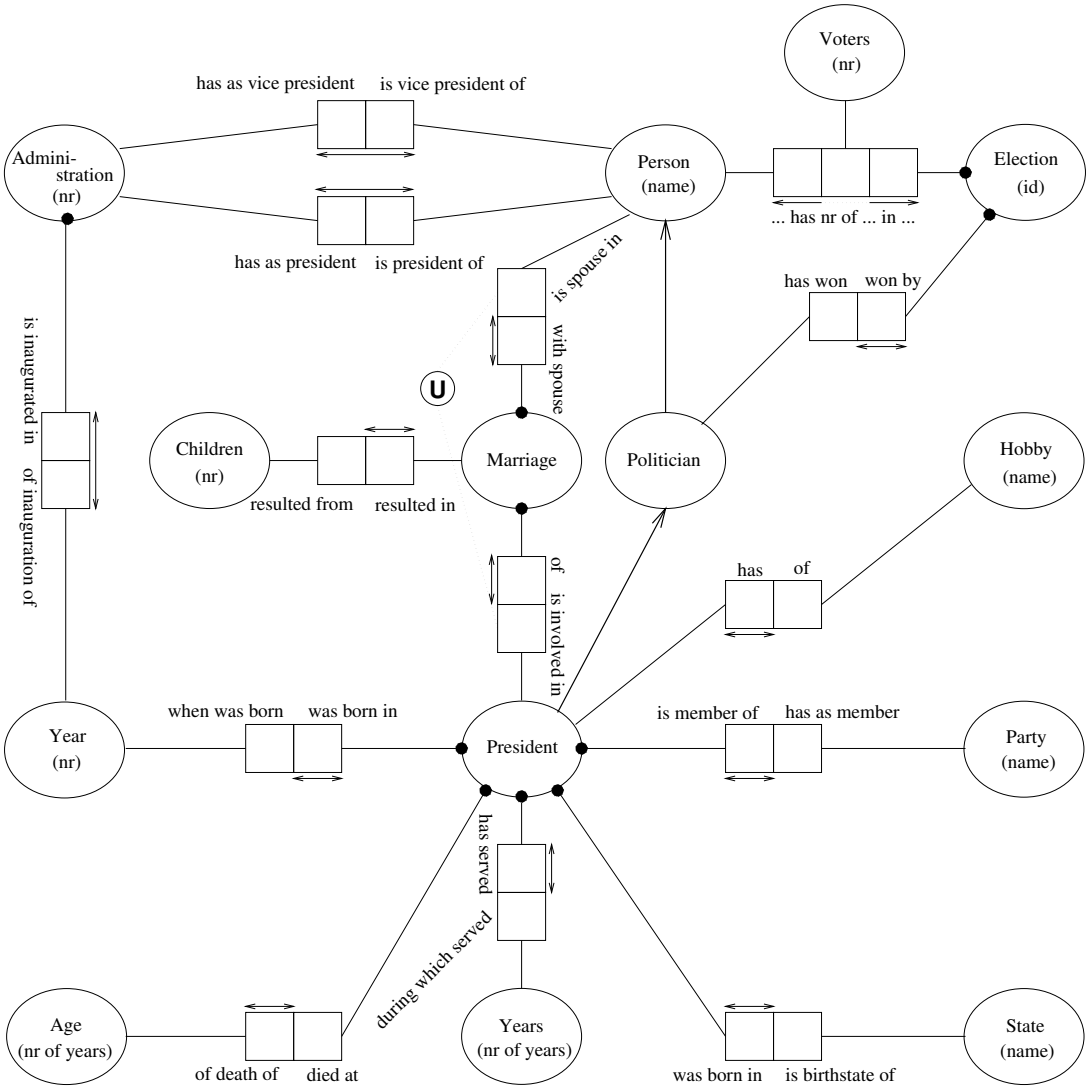

Figure 5: The structure of the presidential database

leads to the node depicted in figure 8. This node shows a second class of associative links. Besides associative links resulting from subtyping, we also distinguish associative links to the reversed formulation of the current focus, i.c. the marriage of a president. When navigating through the hyperindex, refinements to the current focus will take place on the tail of the current focus. By reversing the current focus, refinements can be made on the front as well.

The user decides to select the refinement with as spouse the person, leading to figure 9. The user considers this, for the moment, to be a proper description of the information need. To get an impression of the query result so far, the user selects the beam down option. This results in the node depicted in figure 10. This node is neither part of the hyperbase, nor is it part of the hyperindex. It is an ad-hoc node representing the result of the focus of figure 9 interpreted as a query on the underlying database. The user can now select an instance for further navigation, which will then indeed take place in the hyperbase. Let us presume the user selects the marriage between president Washington and M.D. Custis as starting point for further refinement.

This leads to the node shown in figure 11. This node shows as refinements all information known about M.D. Custis. Since the database only focuses on presidents, there is no other information known about

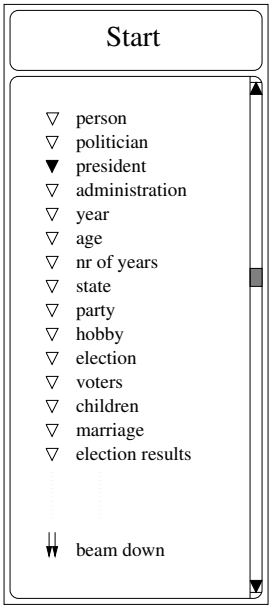

Figure 6: The starting node of the hyperindex

M.D. Custis other than that she is married to president Washington. The only associative link for this node leads to the *objectification* of the marriage instance. In the original ORM schema of figure 5, the object type Marriage is modelled as a so-called *compositely identified object type*. This follows from the constraint pattern and in particular from the inter-relationship uniqueness constraint, depicted as an encircled U. It means that a marriage is identified as a combination of a president and a person. This allows us to treat a combination of a president (Washington) and a person (Custis) as an instance of the object type Marriage.

The user now realises that s/he wanted to know more about Washington's marriage, remembering the initial interest in the number of children of presidential marriages. Therefore, the user now selects the objectification of the marriage as the next point in the navigation. The resulting node, shown in figure 12, shows all information known about this marriage. Since the user was interested in the number of children born in presidential marriages, the user selects the resulted in children refinement. This leads to the node depicted in figure 13. The user is now satisfied with the description of the information need in that the instances (marriage and number of children) are good *examples* of the desired kind of information.

The user could now select the beam up operation to end up at the more general description of the current focus: the marriage that resulted in children. The user can then continue with a beam down to end up with all marriages and the resulting number of children. This latter process of:

1. a beam up of an information object in the hyperbase to a more general description in the hyperindex,
2. followed by a beam down back to the hyperbase that results in all objects with a similar characterisation as the original information object

is an example of *query by example*. Query by example ([Zlo77]) allows users to specify an example of the approximate query they wish to see answered. The system will then try to induce other similar results from this example.

3 The Hyperindex Layer

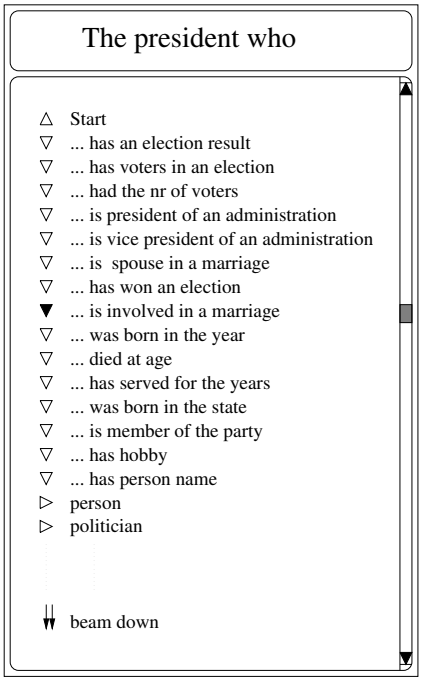

Figure 7: The quest for a president who is married with a politician

To formally define the stratified hypermedia that can be associated to a conceptual schema of an Object-Role Modelling technique, we first need some formal definition of what an ORM conceptual schema is. The formalisation used here is a simplified version of the formalisation of ORM as provided in [HPW93].

3.1 Generalised ORM Schemas

A conceptual schema is build around a set of object types $\mathcal{O}$ and fact types (also called relationship types): $\mathcal{F}$. Each fact type f consists of a set of roles from $\mathcal{P}$. The fact types in $\mathcal{F}$ should form a partition of the roles in $\mathcal{P}$. Due to this partition, the following function for roles can be defined:

$$\mathsf{Fact}(r) = f \iff r \in f$$

All roles have a base, the object type playing the role. This base is provided by the function $\mathsf{Base} : \mathcal{P} \to \mathcal{O}$.

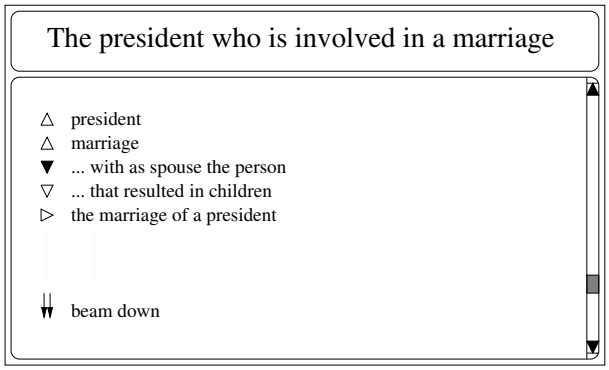

Figure 8: Focus on marriage

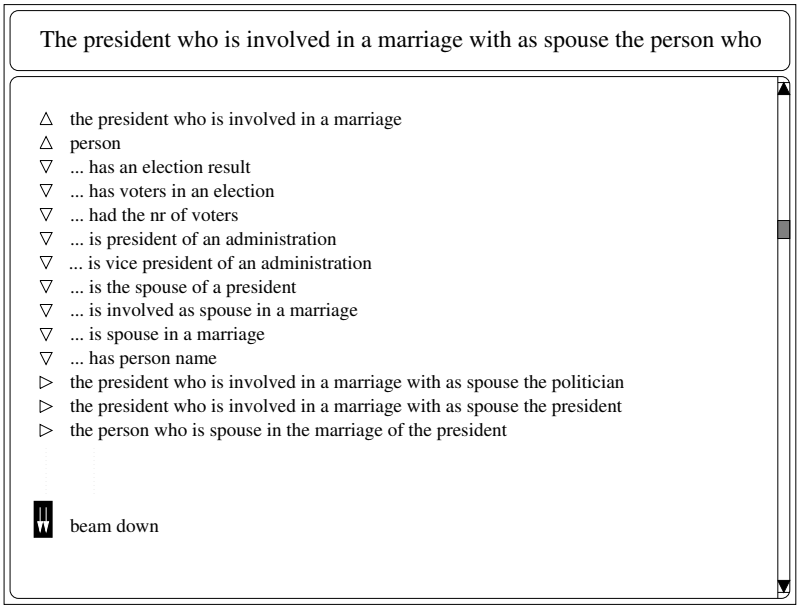

Figure 9: Preliminary result in the hyperindex

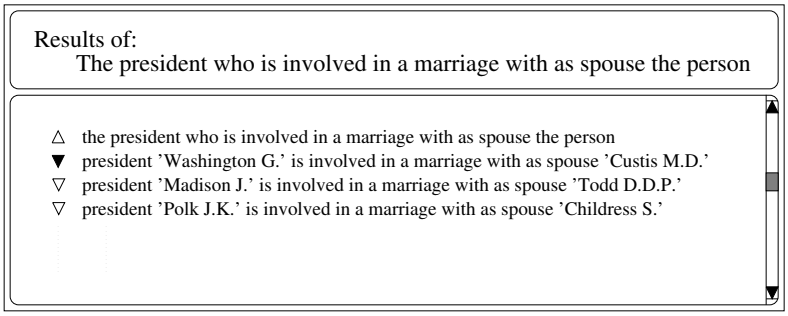

Figure 10: Preliminary result of the query

All Object-Role Modelling variations allow for the definition of type hierarchies. Different ways to introduce such hierarchies exist, e.g. specialisation and generalisation (polymorphism) ([HW93], [HPW93], [PW94], [HP95]). For our purposes we can simply presume the existence of a general notion of a type hierarchy which could involve different flavours of inheritance. The relation $\mathsf{IdfBy} \subseteq \mathcal{O} \times \mathcal{O}$ is used to capture this general notion. The intuition is that if $x \,\mathsf{IdfBy}\, y$, object type x is identified through object type y; which means that x inherits properties from y. It should be clear that this general relation can be used to capture both generalisation (polymorphy) and specialisation relations. The IdfBy relation is presumed to be transitive. Furthermore, we use $x \,\underline{\mathsf{IdfBy}}\, y$ as an abbreviation for $x \,\mathsf{IdfBy}\, y \ \vee \ x = y$, and $x \sim y$ as an abbreviation for $x \,\underline{\mathsf{IdfBy}}\, y \ \vee \ y \,\underline{\mathsf{IdfBy}}\, x$. The $\sim$ relationship is the so-called *type relatedness* relationship. Two types are type related if their populations may share instances.

As an example, consider the ORM schema depicted in figure 14. In this schema we have:

$$\begin{array}{ll} \mathcal{O} = \{A, B, C, D\} & \mathcal{F} = \{f, g\} \\ \mathcal{P} = \{p, q, r, s\} & C \,\mathsf{IdfBy}\, B \\ f = \{p, q\} & g = \{r, s\} \\ \mathsf{Base}(p) = A & \mathsf{Base}(q) = B \\ \mathsf{Base}(r) = C & \mathsf{Base}(s) = D \end{array}$$

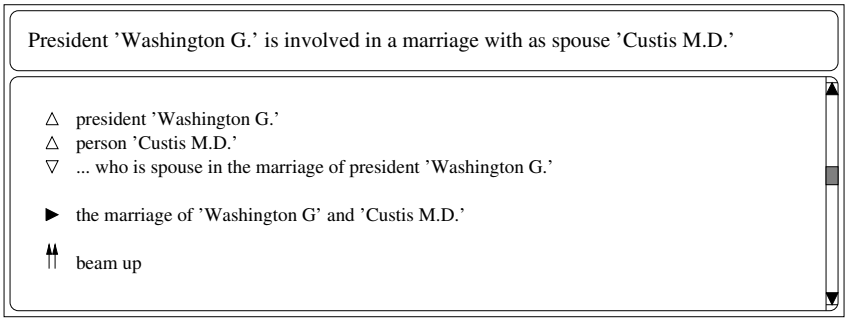

Figure 11: First hyperbase node

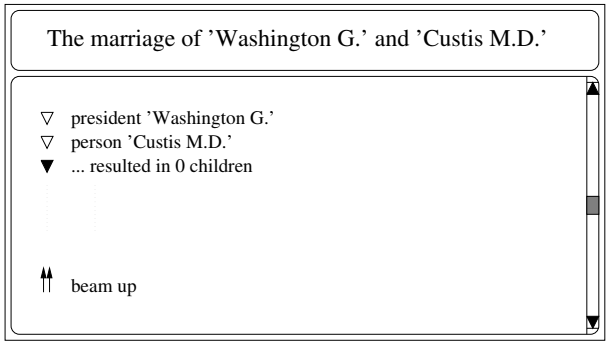

Figure 12: Information about the marriage between Washington and Custis

A further simplification made to the formalisation of ORM models, besides the unified treatment of specialisation and polymorphism, is the treatment of objectification. In ORM modelling (and ER for that matter), one can choose to explicitly objectify fact types. For example, the schema fragment depicted in the left hand side of figure 15 is equivalent to the fragment depicted in the right hand side. For instance, in the presidential database example the object type Marriage could have been modelled conform the left hand side. In the query by navigation mechanism for ORM schemas presented in this paper, objectifications are treated as if they are *flattened*, i.e. as shown in the right hand side of figure 15.

Using this brief formal description of ORM schemas, the stratified hypermedia architecture for ORM models in general (without limiting ourselves to one dialect only) can be build. It should be noted that the formal description equally applies to ER models, so the results of this paper can be translated to ER models

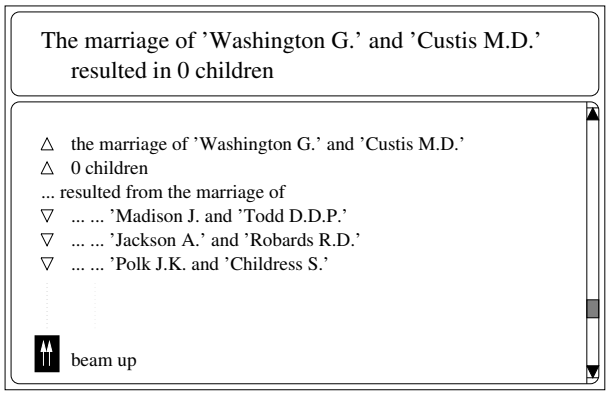

Figure 13: Resulting node

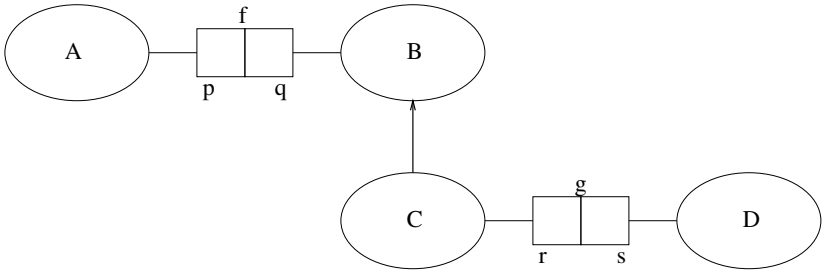

Figure 14: Example ORM Schema

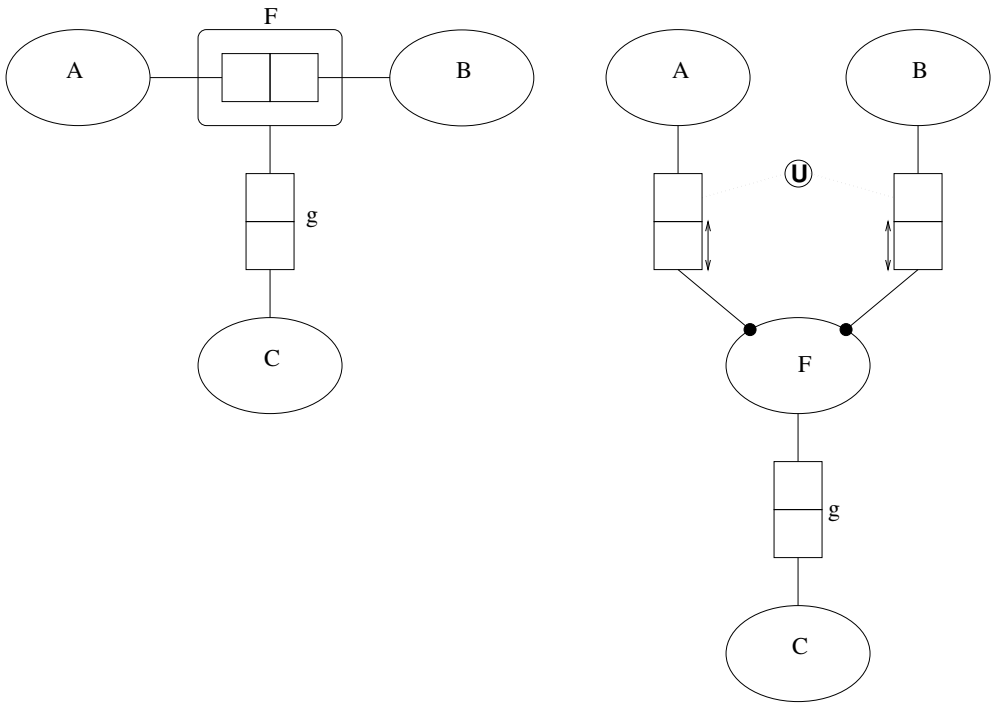

Figure 15: Objectification as an abbreviation

as well. In [BBMP95] the close relationship between ER models and ORM models is discussed in more detail.

3.2 Linear Path Expressions

The backbone of the nodes in the hyperindex and hyperbase is formed by the so-called linear path expressions ([HPW93]). These expressions are build from object types, roles, and instances. All these components can be interpreted as binary relations, and as such concatenated to each other. An object type o occurring in a path expression corresponds to a binary relationship with tuples $\langle x, x \rangle$ for every instance x of type o. A role r corresponds to a binary relationship connecting $\mathsf{Base}(r)$ to $\mathsf{Fact}(r)$, with tuples $\langle x, y \rangle$ where x is the r part of fact instance y. To traverse fact types in a path expression, it must be possible to reverse the order of the $\mathsf{Base}(r)$ and $\mathsf{Fact}(r)$ part of a role. Therefore, $r^{\leftarrow}$ represents the reversed binary relation associated to role r.

When displaying linear path expressions in the nodes of the query by navigation mechanism, the linear path expressions need to be verbalised. These verbalisations can be derived from the names given to the object types and roles from the conceptual schema. In this article we simply presume the existence of a

function ρ verbalising these linear path expressions. For a more detailed discussion on the verbalisation of linear path expressions refer to [Pro94], and [HPW97].

3.3 The Descriptive View Grammar

Formally, a hyperindex is introduced as a structure $\mathsf{L}_1 \triangleq \langle \mathbb{F}_1, \mathbb{N}_1, \mathbb{G}_1, \mathbb{V}_1 \rangle$ (see also [Pro94], [HPW94]). The fragment base ($\mathbb{F}_1$) of the hyperindex simply contains the elements for verbalisations of the linear path expressions. Therefore we should have:

$$\mathbb{F}_1 \quad \subseteq \quad \texttt{Names}^+$$

Some examples from the fragment base of the presidential schema example are:

President, Marriage, involved in, Votes, is vice president of

The following step in defining the hyperindex is the introduction of its schema:

$$\mathbb{G}_1 \quad \triangleq \quad \langle \mathbb{E}_1, \mathbb{P}_1 \rangle$$

In our approach, we navigate through the hyperindex by refinements and enlargements of a linear path expression corresponding to a molecule in the hyperindex. An alternative approach would be to define a grammar for the verbalisation of these path expressions, and navigate through the verbalisations rather than the underlying linear path expression.

The context-free production rules for the hyperindex ($\mathbb{P}_1$), define the way in which linear path expressions can be extended. The grammar $\mathbb{G}_1$ contains for each object type x a corresponding nonterminal (syntactic category) $\langle P_x \rangle$. Instantiations of syntactic category $\langle P_x \rangle$ describe simple properties of (instances of) object type x, i.e., properties that can be derived via a linear path expression starting in object type x. For any $o \in \mathcal{O}$ we have the following rules:

$$\langle S \rangle \;\rightarrow\; \langle P_x \rangle$$
$$\langle P_x \rangle \;\rightarrow\; x$$

The identification hierarchy leads to the following rules. If x IdfBy y, then:

$$\langle P_x \rangle \;\rightarrow\; \langle P_y \rangle$$

which means that properties about y may be used in expressions about x; but *not* vice versa. For roles r and q such that $r \neq q$ and $\mathsf{Fact}(r) = \{r, q\}$ we have:

$$\left\langle P_{\mathsf{Base}(q)} \right\rangle \;\rightarrow\; \left\langle P_{\mathsf{Base}(r)} \right\rangle \circ r \circ \mathsf{Fact}(r) \circ q^{\leftarrow} \circ \mathsf{Base}(q)$$

For the hyperindex we have the following syntactic categories: $\mathbb{E}_1 \triangleq \left\{ \langle P_x \rangle \;\middle|\; x \in \mathcal{O} \right\} \cup \{ \langle S \rangle \}$. Note that the above syntax describes meta-rules, which are concretised by substituting an actual object type for meta-nonterminal x and roles q, r. So, basically this grammar is a two level grammar ([WMP$^+$76]). For the example ORM schema depicted in figure 14 we have:

$$\begin{array}{lll}
\langle S \rangle \rightarrow \langle P_A \rangle & \langle S \rangle \rightarrow \langle P_B \rangle & \langle S \rangle \rightarrow \langle P_C \rangle \\
\langle S \rangle \rightarrow \langle P_D \rangle & \langle P_A \rangle \rightarrow A & \langle P_B \rangle \rightarrow B \\
\langle P_C \rangle \rightarrow C & \langle P_D \rangle \rightarrow D & \langle P_C \rangle \rightarrow \langle P_B \rangle
\end{array}$$

$$\begin{array}{rcl}
\langle P_A \rangle & \rightarrow & \langle P_B \rangle \circ q \circ f \circ p^{\leftarrow} \circ \mathsf{Base}(A) \\
\langle P_B \rangle & \rightarrow & \langle P_A \rangle \circ p \circ f \circ q^{\leftarrow} \circ \mathsf{Base}(B) \\
\langle P_C \rangle & \rightarrow & \langle P_D \rangle \circ r \circ g \circ s^{\leftarrow} \circ \mathsf{Base}(C) \\
\langle P_D \rangle & \rightarrow & \langle P_C \rangle \circ s \circ g \circ r^{\leftarrow} \circ \mathsf{Base}(D)
\end{array}$$

3.4 Molecules in the Index View

In the current approach, the hyperindex for a conceptual schema will contain only a single view. A view is formally introduced as a structure:

$$\mathbb{V}_1 \quad \triangleq \quad \langle \mathbb{S}_1, \omega_1, \mathbb{M}_1, \pi_1, \mathbb{L}_1 \rangle$$

The starting point of this view is $\mathbb{S}_1 \in \mathbb{E}_1$, which is $\langle S \rangle$. The molecules $\mathbb{M}_1$ are formed by the set of linear path expressions augmented with the empty path expression ϵ. Some path expressions can be proven to be structurally empty, i.e. in every population they yield an empty result. As these path expressions are not meaningful in this context, they are omitted from $\mathbb{M}_1$.

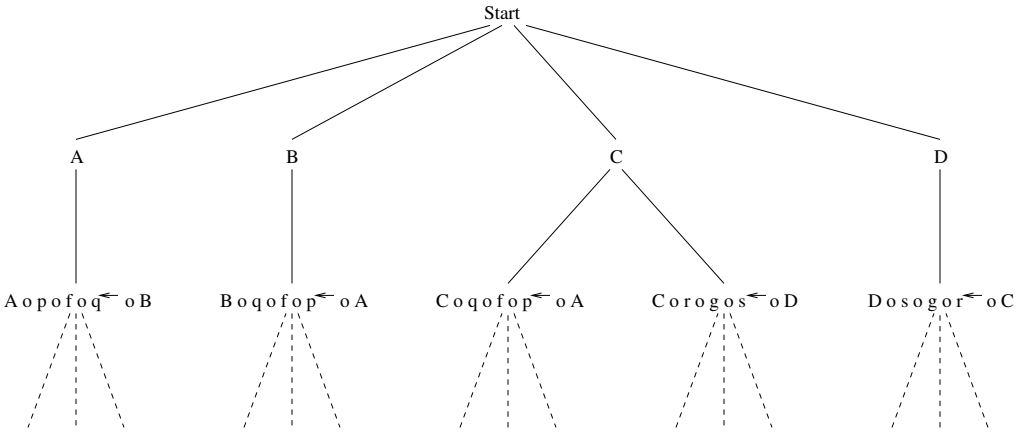

Figure 16: Example hyperindex

The actual structure ω_1 is a subset of $\mathbb{M}_1 \times \mathbb{M}_1$. Let x be an object type, and P be a fragment of a path expression, then this set is identified by the following kinds of structural links:

1. a link from the empty path expression ϵ to any molecule x.
2. a link from a molecule $P\ x$ to a molecule $P\ x \circ r \circ \mathsf{Fact}(r) \circ q^{\leftarrow} \circ \mathsf{Base}(q)$, if $r \neq q$, $\mathsf{Fact}(r) = \{r, q\}$ and x $\underline{\mathsf{IdfBy}}$ $\mathsf{Base}(r)$.

where P is any linear path expression, and q, r are roles and x is an object type. For the example ORM schema in figure 14 this would lead to the (partial) hyperindex as shown in figure 16.

In our running example, as each president is also a politician, associative links are available connecting president with politician in linear path expressions. In order to avoid chaotic structures, these links are only included when such an object type occurs at the end of a linear path expression. So we add an associative link from the president who has as spouse a person to the president who has as spouse a president in figure 9. The front part of path expressions is manipulated only indirectly when navigating through the hyperindex to master the complexity of nodes in this view. To be able to explicitly manipulate the front of path expressions, path reversal is offered. Therefore, we add a link from the president who has as spouse a person to the person who is the spouse of a president in figure 9. We introduce the associative links ($\mathbb{L}_1$) of the hyperindex to cater for the relations in the identification hierarchy, as well as the reversal of the current focus. Let x, y be object types, then we have the following kinds of associative links:

1. a link from a molecule of the form $P\ x$ to a molecule $P\ y$ if $x \sim y$ capturing the identification hierarchy.
2. a link from molecule P to molecule $\mathsf{Rev}(P)$ if $P \neq \mathsf{Rev}(P)$ catering for the reversal of path expressions.

The reversal of a path expressions by Rev is recursively defined as:

$$\begin{aligned} \mathsf{Rev}(P \circ p \circ f \circ q^{\leftarrow} \circ x) &\triangleq x \circ q \circ f \circ p^{\leftarrow} \circ \mathsf{Rev}(P) \\ \mathsf{Rev}(x) &\triangleq x \end{aligned}$$

An example of such a reversal is:

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

In figure 17 we have added some associative links (dotted lines) to the example hyperindex from figure 16.

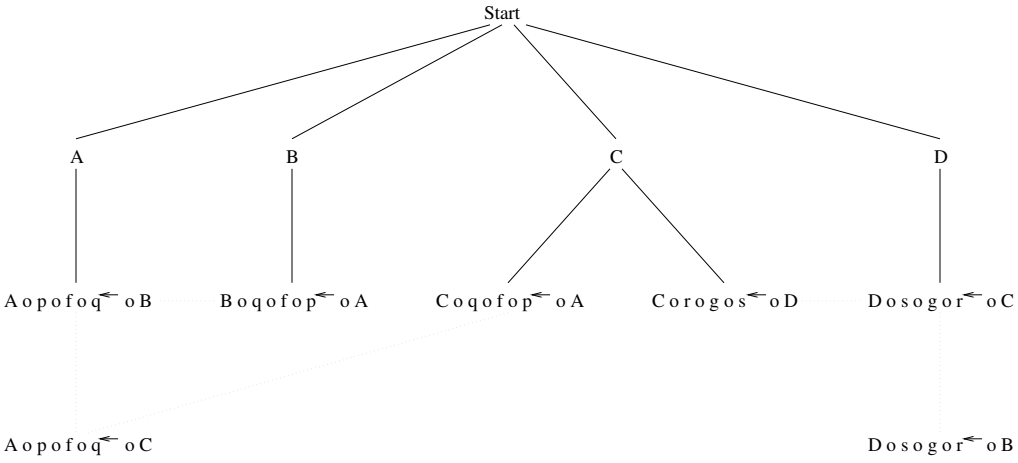

Figure 17: Example hyperindex with associative links

3.5 Presentation of molecules

Molecules are presented by nodes specified by π_1. As stated before, a molecule will be presented by a molecule containing the direct environment of the molecule. The environment of a node is depicted in figure 18. A node, presenting molecule M, is thus made up of:

1. a verbalisation of the molecule itself, identifying the current spot (the focus) in the hyperindex.
2. a verbalisation of each immediate ancestor, showing how to decompose the focus into its components,
3. a verbalisation of each immediate descendant, which suggests how to extend the current focus.
4. a verbalisation of each associated molecule, calling the attention to related alternatives.

The presentation of a molecule is formally identified as:

$$\pi_1(M) \triangleq \langle \rho(M), \rho(\triangle(M)), \rho(\triangledown(M)), \rho(\triangleright(M)) \rangle$$

where the direct environment of M is captured by:

$$\begin{aligned} \triangle(M) &\triangleq \{ N \mid \langle N, M \rangle \in \omega_1 \} \\ \triangledown(M) &\triangleq \{ N \mid \langle M, N \rangle \in \omega_1 \} \\ \triangleright(M) &\triangleq \{ A \mid \langle M, A \rangle \in \mathbb{L}_1 \} \end{aligned}$$

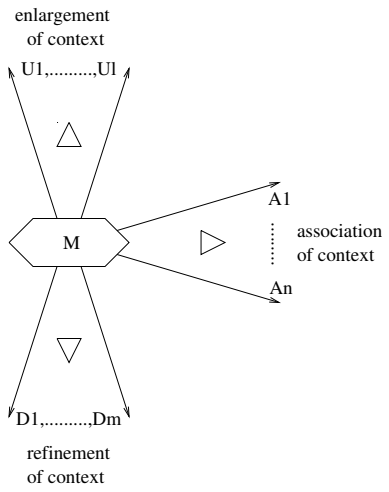

Figure 18: The environment of a molecule

We presume that the verbalisation function ρ is extended to sets of path expressions in the natural way. The general format of a node is displayed in figure 19.

The order in which the elements of the above sets are actually presented in a node can be based on a variety of factors. It can for example be based on the previous search behaviour of a user ([BHW96]). In the context of conceptual schemas, it also makes sense to base this order on conceptual relevance. When looking at a conceptual schema one can distinguish object types that are conceptually more relevant than others. This relevance can be captured as some numerical value expressing the conceptual relevance of some schema component. For instance, the object type President has, intuitively, a higher relevance than Hobby for the modelled domain. In, e.g. [CH94] and [CH93], a procedure is described to determine the most relevant object types in a given conceptual schema. Entries in a node could now be ordered based on the conceptual relevance of the object types occurring in an entry.

Which ordering factors should actually be taken into consideration can only be determined after extensive empirical testing. Currently no implementation is available to provide such a testing environment. In the query tool developed by Asymetrix ([Asy94]) the initial factor for ordering will be conceptual relevance.

All that remains to be done with respect to the presentation of the molecules, is a proper definition of $\rho(P)$ where P is a path expression. This can be done by a set of derivation rules, with an associated preference (using penalty points). As stated before, for a more detailed discussion of such a set of verbalisation rules, refer to [Pro94] or [HPW97].

4 The Hyperbase Layer

In this section we describe the organisation of the hyperbase layer for a conventional information system. This layer is internally organised according to the associated conceptual schema, and is instantiated in accordance with the population of that schema. The stratified hypermedia architecture contains a single view on the population of the associated conceptual schema, the so-called *Base View*. This view describes the complete information base in the format of instantiated linear path expressions. The translation of instantiations into a hyperbase is also carried out bottom-up, the fragment base is defined first, followed by the node base, the schema and the views respectively. As the hyperbase layer is quite similar to the hyperindex layer, we will not provide any additional examples.

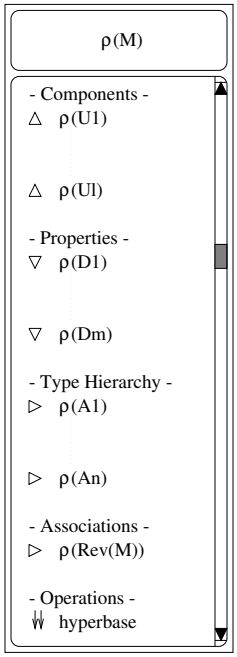

Figure 19: The presentation of a molecule

4.1 Fragment Base

The fragment base ($\mathbb{F}_0$) of the hyperindex is formed from the names used in the verbalisations of the path expressions as is the case for the hyperindex, extended with the denotations of the instances (see e.g. figure 11).

4.2 The Base View Grammar

The grammar rules for the hyperbase ($\mathbb{G}_0$) are formed in the same way as for the hyperindex except for one rule. For any $o \in \mathcal{O}$ and instance i of o we have:

$$\langle P_x \rangle \ \rightarrow \ x \circ i$$

rather than:

$$\langle P_x \rangle \ \rightarrow \ x$$

For the hyperbase we therefore have the same set of syntactic categories: $\mathbb{E}_0 \triangleq \{ \langle P_x \rangle \mid x \in \mathcal{O} \} \cup \{ \langle S \rangle \}$. Similarly to the hyperindex, the starting symbol $\mathbb{S}_0$ is again $\langle S \rangle$. The set of molecules $\mathbb{M}_0$ of the hyperbase corresponds to the path expressions that can be build from the grammar rules in $\mathbb{G}_0$, and which are not empty.

Let x be an object type, let i, j and k be instances, and let P be a fragment of a path expression, then the structure ω_0 of the hyperbase is provided as:

1. a link from the empty path expression ϵ to any molecule $x \circ i$.
2. a link from a molecule $P \ x \circ i$ to a molecule $P \ x \circ i \circ r \circ \mathsf{Fact}(r) \circ j \circ q^{\leftarrow} \circ \mathsf{Base}(q) \circ k$ if $r \neq q$, $\mathsf{Fact}(r) = \{r, q\}$ and $x \ \underline{\mathsf{IdfBy}} \ \mathsf{Base}(r)$.

Note by requiring that the molecules in $\mathbb{M}_0$ correspond to path expressions which do not have an empty result, the instances i, j and k are implicitly required to be associated via the used relationship type $\mathsf{Fact}(r)$.

The associative links ($\mathbb{L}_0$) for the hyperbase are quite similar to the links in the hyperbase. Let x, y be object types and i an instance, then we have the following kinds of associative links:

1. a link from a molecule of the form $P\ x \circ i$ to a molecule $P\ y \circ i$ if $x \sim y$.
2. a link from molecule P to molecule $\mathsf{Rev}(P)$ if $P \neq \mathsf{Rev}(P)$ catering for the reversal of path expressions.

4.3 Presentation of molecules

The presentation of molecules from the hyperbase does not differ from the presentation of molecules from the hyperindex. The verbalisation function ρ should also be able to handle instances in linear path expressions.

5 Relating Hyperbase and Hyperindex

In this section, the hyperbase and hyperindex are related using the beam up and beam down operations. We do this in a top down fashion. First we discuss how hyperbase and hyperindex can be compared to each other presuming some characterisation for the molecules in both layers is provided. This is followed by a discussion of three possible ways of characterising molecules, leading to *strong*, *hybrid*, and *weak* characterisations.

5.1 Inter Layer Navigation

In general, interlayer navigation is a way to operationalise the characterisations of molecules in the (differing) involved layers. In section 2 the characterisation χ of layer L_1 via view V_1 into view V_2 of layer L_2 has been introduced as:

$$\chi \subseteq V_1.\mathbb{M} \times V_2.\mathbb{M}$$

In the remainder of this section this characterisation function is defined in more detail.

For convenience, we assume that each view within any layer is identified by a unique number. The characterisation of view v in terms of view w is denoted as $\chi_{v \to w}$. We number the standard view within the hyperbase layer as view number 0, and the standard view on the hyperindex layer as view number 1. Using characterisations we are able to relate molecules to each other to identify the relevance from one molecule to another. Several ways to determine the similarity between characterisations exist ([Rij75]). A simple and well-known method to compute similarity is the following formula, also well-known as Jaccard's coefficient:

$$\mathtt{Sim}(C_1, C_2) \quad \triangleq \quad \frac{|C_1 \cap C_2|}{|C_1 \cup C_2|}$$

This formula is also useful in the context of multisets. In this case the similarity formula can also be written as:

$$\mathtt{Sim}(C_1, C_2) \quad \triangleq \quad \sum_x \frac{\min(\mathsf{Freq}(x, C_1), \mathsf{Freq}(x, C_2))}{\mathsf{Freq}(x, C_1) + \mathsf{Freq}(x, C_2)}$$

where $\mathsf{Freq}(x, M)$ returns the frequency of element x in multiset M. Multisets are sometimes used to represent the result of queries (SQL, LISA-D). Another advantage is that multisets allow for a more adequate characterisation, as they take frequencies into account.

Inter layer navigation may now be defined from the characterisation relations by using the `Sim` function. The beam operator $\mathsf{Beam}_{n \to m} : \mathbb{M}_n \to (\mathbb{M}_m \to [0, 1])$ associates the molecules from view n with

the molecules from view m in terms of the characterisation function $\chi_{n\to m}$ in terms of the relevance as follows:

$$\mathsf{Beam}_{n\to m}(x) \quad \triangleq \quad \lambda_{y\in\mathbb{M}_m}.\mathtt{Sim}(\chi_{n\to m}(x), \chi_{m\to m}(y))$$

By taking $n = m$ *intra* layer beaming results, and by taking $n \neq m$ *inter* layer beaming results. Beaming from a lower numbered view to a higher number view is called *beam up*, the opposite direction is known as *beam down*. Since this article only considers two levels in the stratified hypermedia architecture the n and m are limited to 0 for the hyperbase, and 1 for the hyperindex.

5.2 Characterisations

As stated before, for the actual characterisation functions we distinguish between three different flavours which are introduced in the remainder of this section. The first kind, *strong characterisation*, uses de-instantiation of path expressions to obtain characterisations. The second kind uses the presentation, the nodes, as a basis for the characterisation of molecules, leading to *weak characterisations*. The third class of beam operations forms a hybrid between the first two classes. This flavour bases itself on the components of the path expressions representing the molecules, leading to the *hybrid characterisation*.

Which characterisation works best in practice needs to be established by testing. However, it might turn out that users would like to have explicit control on each of these characterisation mechanisms and combine the similarity measures between two molecules using weights.

5.2.1 Strong Characterisation

The strong characterisation of a molecule (instantiated path expression) from the hyperbase in terms of molecules (path expressions) from the hyperindex consists of all queries that (1) are structurally compatible with the molecule to be characterised, and (2) from which this molecule may result in the query result. The characterisation is effectively obtained by the replacement of all typed instances by all types which are associated to these instances.

Let x be an object type, i an instance, p a role, and P, Q linear path expressions, then the strong characterisation for the hyperbase is recursively defined as:

$$\begin{aligned}
\chi^s_{0\to 1}(x \circ i) &\triangleq \{\!\!\{ y\!\uparrow^1 \mid x \sim y \}\!\!\} \\
\chi^s_{0\to 1}(p) &\triangleq \{\!\!\{ p \}\!\!\} \\
\chi^s_{0\to 1}(p^{\leftarrow}) &\triangleq \{\!\!\{ p^{\leftarrow} \}\!\!\} \\
\chi^s_{0\to 1}(P \circ Q) &\triangleq \{\!\!\{ A \circ B\!\uparrow^{n\times m} \mid A \in^n \chi^s_{0\to 1}(P) \wedge B \in^m \chi^s_{0\to 1}(Q) \wedge \mathsf{InPop}(P \circ Q, A \circ B) \}\!\!\}
\end{aligned}$$

where InPop is defined as:

$$\mathsf{InPop}(A, B) \triangleq \mu[\![A]\!]\,(\mathsf{Pop}) \subseteq \mu[\![B]\!]\,(\mathsf{Pop})$$

This limits the $A \circ B$ combinations to those that define a query that is a superset of the original query $P \circ Q$. The expression $P \circ Q$ contains instances whereas $A \circ B$ contains none. This means that $P \circ Q$ is (should be) more limiting than $A \circ B$. Finally, expression $x\!\uparrow^n$ refers to x with frequency n, whereas $x \in^n M$ is used to denote that x occurs in multiset M with frequency n.

Please note again that since we limit ourselves to only two layers (hyperbase and hyperindex), the characterisation is only provided for navigation between these two layers.

For the hyperindex, we have the characterisation:

$$\chi^s_{1\to 1}(x \circ i) \quad \triangleq \quad \{\!\!\{ y\!\uparrow^1 \mid x \sim y \}\!\!\}$$

$$
\begin{aligned}
\chi^s_{1\to 1}(p) &\triangleq \{\!\![p]\!\!\} \\
\chi^s_{1\to 1}(p^{\leftarrow}) &\triangleq \{\!\![p^{\leftarrow}]\!\!\} \\
\chi^s_{1\to 1}(P \circ Q) &\triangleq \{\!\![A \circ B\uparrow^{n\times m} \mid A \in^n \chi^s_{1\to 1}(P) \wedge B \in^m \chi^s_{1\to 1}(Q) \wedge \mathsf{InPop}(P \circ Q, A \circ B)]\!\!\}
\end{aligned}
$$

As an example, the path expression $B \circ i \circ p \circ f \circ j \circ q^{\leftarrow} \circ C \circ k$ from the hyperbase of the schema in figure 20 yields the following characterisation: $\{\![A \circ p \circ f \circ q^{\leftarrow} \circ C, B \circ p \circ f \circ q^{\leftarrow} \circ C]\!\}$. This set is also exactly the characterisation of the path expression $A \circ p \circ f \circ q^{\leftarrow} \circ C$ from the hyperindex that can be associated to schema 20.

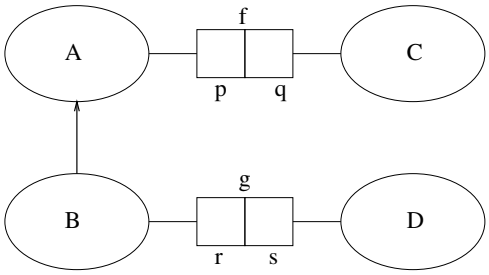

Figure 20: Example characterisations

A possible way to enhance this flavour of characterisation, is to add a conceptual relevance value to each of the elements contained in a characterisation. For a given expression in a characterisation, the conceptual relevance can be defined as the average conceptual relevance of all types and roles occurring in this expression.

5.2.2 Hybrid Characterisation

The hybrid characterisation of a molecule describes an instantiated path expression in terms of the frequencies of all roles, reverse roles, and object types in this expression. It is similar to the weighted vector model for conventional documents. A difference however is that type related object types are also taken into account in the characterisation. For the hyperbase we therefore have:

$$
\begin{aligned}
\chi^h_{0\to 1}(x \circ i) &\triangleq \{\!\![y\uparrow^1 \mid x \sim y]\!\!\} \\
\chi^h_{0\to 1}(p) &\triangleq \{\!\![p]\!\!\} \\
\chi^h_{0\to 1}(p^{\leftarrow}) &\triangleq \{\!\![p^{\leftarrow}]\!\!\} \\
\chi^h_{0\to 1}(P \circ Q) &\triangleq \chi^h_{0\to 1}(P) \cup \chi^h_{0\to 1}(Q)
\end{aligned}
$$

For the hyperindex this leads to:

$$
\begin{aligned}
\chi^h_{1\to 1}(x \circ i) &\triangleq \{\!\![y\uparrow^1 \mid x \sim y]\!\!\} \\
\chi^h_{1\to 1}(p) &\triangleq \{\!\![p]\!\!\} \\
\chi^h_{1\to 1}(p^{\leftarrow}) &\triangleq \{\!\![p^{\leftarrow}]\!\!\} \\
\chi^h_{1\to 1}(P \circ Q) &\triangleq \chi^h_{1\to 1}(P) \cup \chi^h_{1\to 1}(Q)
\end{aligned}
$$

For the example path expression from the hyperbase ($B \circ i \circ p \circ f \circ j \circ q^{\leftarrow} \circ C$), this leads to the following result: $\{\![A, B, p, f, q^{\leftarrow}, C]\!\}$. The hybrid characterisation of the (hyperindex) path expression: $A \circ p \circ f \circ q^{\leftarrow} \circ C$ is also: $\{\![A, B, p, f, q^{\leftarrow}, C]\!\}$.

This hybrid characterisation flavour could also be enhanced by using conceptual relevance factors for the components.

5.2.3 Weak Characterisation

Both strong and hybrid characterisation are based on the structure of instantiated path expressions. Instantiated path expressions are abstract objects, that are concretised by their representation to the user. In the stratified hypermedia architecture, this representation is captured by the presentation function π. This provides the opportunity to make a characterisation of instantiated path expressions based on their presentation. Presentations of instantiated path expressions are constructed from verbalisations of structural elements. These names are exploited, and monitored (with their frequency) in the weak characterisation:

$$\chi^{w}_{1\to 1}(M) \quad \triangleq \quad \{\!\!\{ w\!\uparrow^{n} \mid w \in \texttt{Names} \wedge w\ \mathsf{OccursIn}_n\ \rho(M) \}\!\!\}$$

where $x\ \mathsf{OccursIn}_n\ y$ is a predicate expressing the occurrence frequency (n) of a string x in a string y.

As an example of characterisations based on the presentation of molecules, consider:

$$\begin{aligned}
&\chi^{w}_{1\to 1}(\text{"president is involved in marriage with politician"}) \\
&\quad = \{\!\!\{ [\mathsf{is}, \mathsf{involved}, \mathsf{in}, \mathsf{marriage}, \mathsf{politician}, \mathsf{president}, \mathsf{with}] \}\!\!\} \\
&\chi^{w}_{1\to 1}(\text{"president was born in year"}) \\
&\quad = \{\!\!\{ [\mathsf{born}, \mathsf{was}, \mathsf{in}, \mathsf{president}, \mathsf{year}] \}\!\!\}
\end{aligned}$$

$$\begin{aligned}
&\chi^{w}_{0\to 1}(\text{"president 'J.F. Kennedy' was born in year 1917, and died at age 46"}) \\
&\quad = \{\!\!\{ [\mathsf{age}, \mathsf{at}, \mathsf{born}, \mathsf{died}, \mathsf{was}, \mathsf{at}, \mathsf{and}, \mathsf{in}, \mathsf{president}, \mathsf{year}] \}\!\!\}
\end{aligned}$$

where "president involved in marriage with politician" denotes the path expression (molecule) with verbalisation president involved in marriage with politician. For the example, we have the following relevance:

$$\begin{aligned}
&\texttt{Sim}(\chi^{w}_{1\to 1}(\text{"president was born in year"}), \\
&\quad \chi^{w}_{0\to 1}(\text{"president 'J.F. Kennedy' was born in year 1917, and died at age 46"})) \\
&= \frac{\left|\{\!\!\{ [\mathsf{born}, \mathsf{was}, \mathsf{in}, \mathsf{president}, \mathsf{year}] \}\!\!\}\right|}{\left|\left\{\!\!\left\{ \begin{bmatrix} \mathsf{age}, \mathsf{at}, \mathsf{born}, \mathsf{born}, \mathsf{died}, \mathsf{was}, \mathsf{was}, \mathsf{at}, \\ \mathsf{and}, \mathsf{in}, \mathsf{in}, \mathsf{president}, \mathsf{president}, \mathsf{year}, \mathsf{year} \end{bmatrix} \right\}\!\!\right\}\right|} \\
&= \tfrac{5}{15} = 1/3 \approx 0.33
\end{aligned}$$

A possible refinement for the weak characterisation of molecules would be the introduction of stop and stemming lists, yielding a more refined base for characterisation.

For weak characterisations, conceptual relevance makes less sense as it is a presentation based characterisation rather than a contents based one.

6 Query By Construction

The queries resulting from the explorative phase, supported by the query by navigation mechanism, are rather simple in that they always correspond to linear path expressions. This means that the resulting queries will never contain operations like *union*, *intersection*, *difference*, etc. Therefore we propose to integrate the query by navigation interface with a LISA-D structure editor.

When such a structure editor for queries is added the linear path expressions resulting from query by navigation can be combined into more complex ones, utilising the expressive power of LISA-D to its fullest. For a complete discussion of all possible operations in LISA-D please refer to [HPW93], [PW95] and [HPW97]. An example session of such an editor is depicted in figure 21. The depicted query is a formulation in LISA-D of the request:

List the parties which have a member who was a president, was born in Oregon, has model railroads as a hobby, and was never vice president.

The process of building a query by combining linear path expressions in such a structure editor, is referred to as *query by construction*. In this example we can see four linear path expressions:

party which has as president,

president who was born in the state,

president who has the hobby,

president who is the vice president of an administration

To create a more readable verbalisation, the President object type has been removed from the last three linear paths. This is merely a verbalisation issue though. Besides these four linear paths, this query contains the two operations: AND ALSO and BUT NOT, and two constants: Oregon, Model railroads. The two operations correspond to multi-set intersection and multi-set difference respectively. It is quite possible to develop a direct manupulation interface ([Sch83]) which allows users to manipulate these query components, while the system provides guidance based on the grammar of LISA-D. For other examples of graphical query interfaces, refer to e.g. [ADD+92], [Ros94].

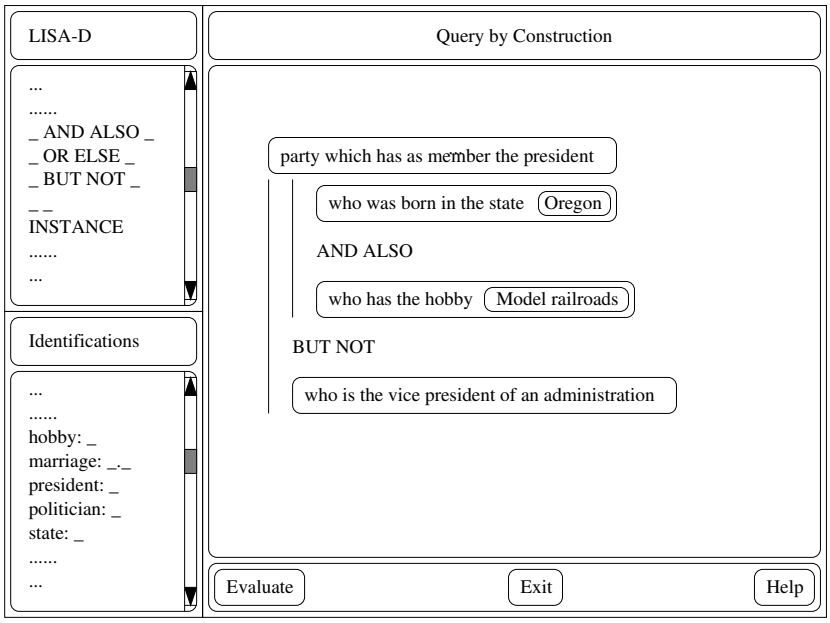

Figure 21: Query by construction

The presentation of query results can be fully integrated with the query by navigation interface. A query result can be used as a starting point for a navigation session through the hyperbase. When evaluating a LISA-D query a (multi) set of binary tuples results. To this result the following set of molecules in the hyperbase can be associated:

$$\mathsf{Result}(Q) \triangleq \{A \mid \mathsf{InPop}(A, \mathbb{D}[\![Q]\!])\}$$

where $\mathbb{D}[\![Q]\!]$ denotes the path expression associated to LISA-D query Q. These results can then be presented by means of a (virtual) molecule as presented in figure 22, from which instances can be selected for a further exploration of the hyperbase.

For a formal treatment of LISA-D, refer to [HPW93] or [HPW97]. The expressiveness of the LISA-D query language is high. As an example (taken from [HPW97]), consider the use of recursive macros in the

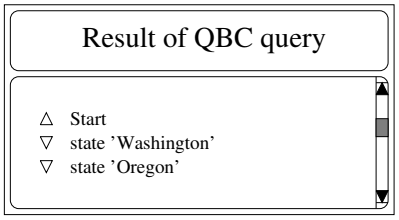

Figure 22: The result of a query

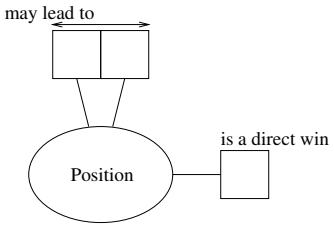

Figure 23: Schema of a simple game

context of a conceptual schema for a 2-person game as illustrated in figure 23. In this domain the fact type may lead to describes how positions can be reached from one another. The unary fact type is a direct win gives all winning positions for the first player. The question now is to yield all positions from which the first player can win ([Cha88]). This is captured by the following macro:

```
Winning Positions IS is direct win UNION may lead to Position ALL IN Position may lead to Winning Positions
```

The above query is an example of a query that cannot be expressed as a so-called *stratified query* (see figure 24, taken from [Cha88]). Stratified queries can express all the first order queries and negation is allowed between the so-called *strata*. It has been shown however that stratified queries do not express all fixpoint queries, in particular, they have difficulty taking fixpoints over universal quantifiers, such as is needed in the above query (see [Kol91] and [Dah87]).

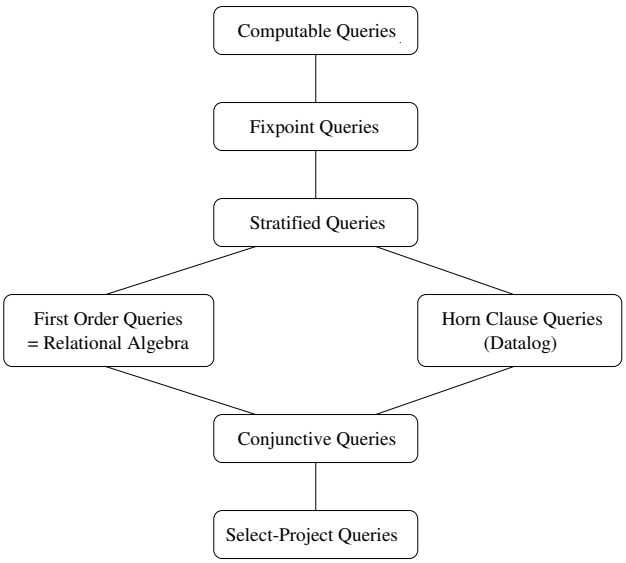

Figure 24: Query language expressiveness hierarchy

The macro mechanism of LISA-D allows for the specification of arbitrary fixpoint queries. There are,

however, some relatively simple queries which cannot be expressed as fixpoint queries (due to lack of arithmetic operations). An example is the query *Even*, which determines whether the number of instances in a certain relation is even ([Cha88]). LISA-D provides the necessary basic arithmetic operators and this query can therefore be straightforwardly expressed:

EVEN R IS NUMBER OF(R) MOD 2 = 0

Whether every computable query can be expressed in LISA-D remains an issue for further research.

7 Conclusions

In this paper a new approach to query formulation support in the context of information systems has been introduced. The query by navigation system can be used to formulate so-called linear path expressions in the conceptual query language LISA-D. This mechanism allows a user to navigate through the conceptual schema of an information system as well as the population of that system. The linear paths resulting from a query by navigation session can be combined into more complex queries using a syntax directed editor for query by construction. The query language LISA-D, which has a high expressiveness, thus becomes an intuitive mechanism for the formulation of queries.

As a next step, the proposed querying system should actually be implemented after which empirical testing can provide the feedback needed to make proper decisions on how to configure and tune the navigation system. Since in real life applications the size of conceptual schemas may quite well match that of wall paper, users may still become lost in conceptual space when using a two level query by navigation mechanism as presented here. In such cases, even a query by navigation system may fail to prevent the user from getting lost in conceptual space. We will therefore also employ the existing idea ([CH93], [CH94]) of defining abstractions on information structures to derive a multi level query by navigation mechanism.

Acknowledgements

We would like to thank the anonymous referees for their comments and suggestions, which have led to improvements of the original article.

References

[AAC+89] M. Agosti, A. Archi, R. Colotti, R.M. Di Giorgi, G. Gradenigo, B. Inghirami, P. Matiello, R. Nannuci, and M. Ragona. New prospectives in information retrieval techniques: a hypertext prototype in environmental law. In *Online Management 89, Proceedings 13th International Online Information*, pages 483–494, London, United Kingdom, 1989.

[ACG91] M. Agosti, R. Colotti, and G. Gradenigo. A two-level hypertext retrieval model for legal data. In A. Bookstein, Y. Chiaramella, G. Salton, and V.V. Raghavan, editors, *Proceedings of the 14th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 316–325, Chicago, Illinois, October 1991. ACM Press.

[ADD+92] A. Auddino, Y. Dennebouy, Y Dupont, E. Fontana, S. Spaccapietra, and Z. Tari. SUPER - Visual Interaction with an Object-based ER Model. In G. Pernul and A.M. Tjoa, editors, *11th International Conference on the Entity-Relationship Approach*, volume 340–356 of *Lecture Notes in Computer Science*, pages 423–439. Springer-Verlag, 1992.

[Asy94] Asymetrix. *InfoModeler User Manual*. Asymetrix Corporation, 110-110th Avenue NE, Suite 700, Bellevue, WA 98004, Washington, 1994.

[BBB91] R. Bosman, R. Bouwman, and P.D. Bruza. The Effectiveness of Navigable Information Disclosure Systems. In G.A.M. Kempen, editor, *Proceedings of the Informatiewetenschap 1991 conference*, Nijmegen, The Netherlands, 1991.

[BBMP95] G.H.W.M. Bronts, S.J. Brouwer, C.L.J. Martens, and H.A. Proper. A Unifying Object Role Modelling Approach. *Information Systems*, 20(3):213–235, 1995.

[BHW96] F.C. Berger, A.H.M. ter Hofstede, and Th.P. van der Weide. Supporting Query by Navigation. In R. Leon, editor, *Information retrieval: New systems and current research, Proceedings of the 16th Research Colloquium of the British Computer Society Information Retrieval Specialists Group*, pages 26–46, Drymen, Scotland, EU, 1996. Taylor Graham.

[Bru90] Peter D. Bruza. Hyperindices: A novel aid for searching in hypermedia. In A. Rizk, N. Streitz, and J. Andre, editors, *Proceedings of the European Conference on Hypertext - ECHT 90*, pages 109–122, Cambridge, United Kingdo, 1990. Cambridge University Press.

[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.

[CD90] W.B. Croft and R. Das. Experiments with Query Acquisition and Use in Document Retrieval Systems. In J. Vidick, editor, *Proceedings of the 13th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 349–368. ACM Press, 1990.

[CH93] L.J. Campbell and T.A. Halpin. Automated Support for Conceptual to External Mapping. In S. Brinkkemper and F. Harmsen, editors, *Proceedings of the Fourth Workshop on the Next Generation of CASE Tools*, pages 35–51, Paris, France, June 1993.

[CH94] L.J. Campbell and T.A. Halpin. Abstraction Techniques for Conceptual Schemas. In R. Sacks-Davis, editor, *Proceedings of the 5th Australasian Database Conference*, volume 16, pages 374–388, Christchurch, New Zealand, January 1994. Global Publications Services.

[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.

[Cra78] T. Craven. Linked phrase indexing. *Information Processing & Management*, 14(6):469–476, 1978.

[Dah87] E. Dahlhaus. Skolem normal forms concerning the least fixpoint. In E. Borger, editor, *Computation Theory and Logic*, volume 270 of *Lecture Notes in Computer Science*, pages 101–106. Springer-Verlag, 1987.

[Far80] J. Farradane. Relational indexing part II. *Journal of Information Science*, 1(6):313–324, 1980.

[GGP89] R. Godin, J. Gecsei, and C. Pichet. Design of a Browsing Interface for Information Retrieval. In N.J. Belkin and C.J. van Rijsbergen, editors, *Proceedings of the 12th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 32–37, Cambridge, Massachusetts, June 1989. ACM Press.

[Hal95] T.A. Halpin. *Conceptual Schema and Relational Database Design*. Prentice-Hall, Sydney, Australia, 2nd edition, 1995.

[HM94] T.A. Halpin and R. Meersman, editors. *Proceedings of the First International Conference on Object-Role Modelling (ORM-1)*. Key Centre for Software Technology, University of Queensland, Brisbane, Australia, Magnetic Island, Australia, July 1994.

[HP95] T.A. Halpin and H.A. Proper. Subtyping and Polymorphism in Object-Role Modelling. *Data & Knowledge Engineering*, 15:251–281, 1995.

[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.

[HPW94] 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

[HPW97] A.H.M. ter Hofstede, H.A. Proper, and Th.P. van der Weide. Exploiting Fact Verbalisation in Conceptual Information Modelling. *Information Systems*, 22(6/7):349–385, September 1997.

[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.

[Kol91] P.G. Kolaitis. The expressive power of stratified logic programs. *Information and Computation*, 90(1):50–66, January 1991.

[Luc90] D. Lucarella. A Model for Hypertext-Based Information Retrieval. In *Proceedings of the European Conference on Hypertext - ECHT 90*, pages 81–94, Cambridge, United Kingdom, 1990. Cambridge University Press.

[Mar77] M.E. Maron. On Indexing, Retrieval and the Meaning of About. *Journal of the American Society for Information Science*, 28(1):38–43, 1977.

[Mee82] R. Meersman. The RIDL Conceptual Language. Research report, International Centre for Information Analysis Services, Control Data Belgium, Inc., Brussels, Belgium, 1982.

[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

[Pol93] R. Pollard. A Hypertext-based thesaurus as a subject browsing aid for bibliographic databases. *Information Processing & Management*, 29(3):345–358, 1993.

[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.

[Rij75] C.J. van Rijsbergen. *Information Retrieval*. Butterworths, London, United Kingdom, 1975.

[Rij86] C.J. van Rijsbergen. A New Theoretical Framework for Information Retrieval. In *Proceedings of the 9th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 194–200. ACM Press, 1986.

[Rij89] C. J. van Rijsbergen. Towards an information logic. In *Proceedings of the 12th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 77–86, Cambridge, Massachusetts, United States, June 1989. ACM Press.

[Rij90] C.J. van Rijsbergen. The Science of Information Retrieval: Its Methodology and Logic, 1990. Proceedings of the European Summer School in Information Retrieval.

[Ros94] P. Rosengren. Using Visual ER Query Systems in Real World Applications. In G.M. Wijers, S. Brinkkemper, and T. Wasserman, editors, *Proceedings of the Sixth International Conference CAiSE'94 on Advanced Information Systems Engineering*, volume 811 of *Lecture Notes in Computer Science*, pages 394–405, Utrecht, The Netherlands, June 1994. Springer-Verlag.

[RWB93] R. Rada, W. Wang, and A. Birchall. Retrieval Hierarchies in Hypertext. *Information Processing & Management*, 29(3):359–372, 1993.

[Sal89] G. Salton. *Automatic Text Processing–The Transformation, Analysis, and Retrieval of Information by Computer*. Addison-Wesley, Reading, Massachusetts, 1989.

[Sch83] B. Schneiderman. Direct Manipulation: A Step Beyond Programming Languages. *IEEE Computer*, 16(8):57–69, 1983.

[SDBW91] P.L. van der Spiegel, J.T.W. Driessen, P.D. Bruza, and Th.P. van der Weide. A Transaction Model for Hypertext. In D. Karagiannis, editor, *Proceedings of the Data Base and Expert System Applications Conference (DEXA 91)*, pages 281–286, Berlin, Germany, 1991. Springer-Verlag.

[SM83] G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill New York, NY, 1983.

[Win90] J.J.V.R. Wintraecken. *The NIAM Information Analysis Method: Theory and Practice*. Kluwer, Deventer, The Netherlands, EU, 1990.

[WMP+76] A. van Wijngaarden, B.J. Mailloux, J.E.L. Peck, C.H.A. Koster, M. Sintzoff, C.H. Lindsey, L.T. Meertens, and R.G. Fisker. *Revised Report on the Algorithmic Language ALGOL 68*. Springer-Verlag, Berlin, Germany, 1976.

[Zlo77] M.M. Zloof. Query-by-Example: a data base language. *IBM Systems Journal*, 16(4):324–343, 1977.