Exploring Fact Verbalisations for Conceptual Query Formulation

A.H.M. ter Hofstede[1] and H.A. Proper[2,3] and Th.P. van der Weide[4]

PUBLISHED AS:

A.H.M. ter Hofstede, H.A. Proper, and Th.P. van der Weide. Exploring Fact Verbalisations for Conceptual Query Formulation. In R.P. van de Riet, J.F.M. Burg, and A.J. van der Vos, editors, *Proceedings of the Second International Workshop on Applications of Natural Language to Databases (NLDB'96)*, pages 40–51, Amsterdam, The Netherlands, EU, June 1996. IOS Press, Amsterdam, The Netherlands, EU. ISBN 9051992734

Abstract. An increasing number of conceptual modelling techniques use verbalisation of sample data to derive a model for the given universe of discourse (the problem domain). The underlying assumption is that by elaborate verbalisation of samples taken from the universe of discourse one can elicit an overview of the concepts relevant for the universe of discourse and their interrelationships. In each of these approaches, information modelling is considered as a process leading to some form of an information grammar that describes the expert language; the language used to communicate in the universe of discourse.

In this article we start by discussing a mechanism to support these verbalisations and associated information grammar. We then show how to mine the richness of the verbalisations in the context of a conceptual query language. It is shown how these verbalisations lead to formulations that closely resemble the expert language.

1 Introduction

When developing an information system using a conceptual modelling approach, usually the first task is to derive a complete model of the underlying universe of discourse. A wide range of modern conceptual modelling techniques initiate this task by verbalising sample forms, cases, etc. taken from the universe of discourse. This verbalisation process is usually conducted in close cooperation with domain experts. Examples of such approaches are the

[1]Department of Computer Science, The University of Queensland, Brisbane, Australia 4072,

[2]Cooperative Information Systems Research Centre, Faculty of Information Technology, Queensland University of Technology, GPO Box 2434, Brisbane, 4001 Australia, E.Proper@acm.org

[3]Research Discovery Unit, Research Data Network Cooperative Research Centre, The University of Queensland, Level 7, Gehrmann Laboratories, Brisbane, 4072 Australia

[4]Computing Science Institute, University of Nijmegen, Toernooiveld, 6525 ED Nijmegen, The Netherlands

ER variation discussed in [BCDT95], NIAM and Object-Role Modelling (ORM) ([Hal95]), OMT ([RBP+91]), and the KISS method ([Kri94]).

In recent years, research has been conducted where principles from linguistics are applied to improve the underlying theoretic foundations of these verbalisation based approaches ([DR91], [BR95], [BCDT95]). The underlying assumption is that by elaborate verbalisation of the samples from the universe of discourse, a complete overview of the structure and rules in the universe of discourse can be obtained. The verbalisations resulting from this initial step are consequently used as input for the actual design of the conceptual model.

In this article we only focus on information modelling aspects of conceptual modelling. Therefore, we use verbalisations to derive the so called *information model. In the next section we argue that an information model essentially provides a crude grammar of the language spoken by the communication partners in the universe of discourse (the expert language). The challenge presented in this article is to re-use the verbalisations in the formulations of queries, thus bringing the resulting query language closer to the language of the users.*

This challenge was born out of frustration with the formulation of queries in SQL. After all the trouble modellers and users have gone through in verbalising samples from the universe of discourse, it comes as an anti-climax if users (and database administrators) still have to use SQL to formulate queries, and even more, have to refer to the actual tables to access the information. This frustration has fueled the development of so-called conceptual query languages such as RIDL ([Mee82]) and LISA-D ([HPW93]).

The focus of this paper is both on a discussion of verbalisation of samples provided during the analyses phase, as well as on the application of these verbalisations in the context of a conceptual query language. The structure of the paper is as follows. In section 2, the basic philosophy of verbalisation based modelling techniques is addressed, with a special focus on Object-Role Modelling (ORM). Section 3 then discusses how an information grammar, and associated query language, is derived from these verbalisations. This is followed in section 4 by a brief discussion of a way to support users in query formulation processes, and the use of such a conceptual query language as a feedback mechanism for natural language query interfaces. For a more elaborate, and formal, treatment of the ideas discussed in this paper, refer to [HPW97].

2 Fact Verbalisation as a Basic Philosophy

The point of departure for Object Role Modelling (ORM) is a description of the communication in the universe of discourse in terms of a set of sample sentences ([Hal95]). The effect is that an information system does not represent the objects of the real world but rather the *facts known about these objects. The facts themselves are composed of references to these objects. This is what earned ORM the predicate* fact oriented modelling. Objects are referenced by their properties. A fundamental principle of ORM is therefore *fact orientation, stating that each state of a universe of discourse can be fully described (characterised) by facts. From these facts the fact types (in ER terminology: relationship types) can be derived.*

An important issue for information modelling is the balance between completeness and the absence of ambiguities and redundancy. Therefore, samples need to be restricted to what is referred to as ORM Normalform. In [Hal95], this normal form and its rationale is exposed in full detail. The restriction to normal form enforces the use of elementary sentences and requires values to be fully typed. As a result, each of these elementary sen-

tences represents a valid sentence, a *fact formulation, in the expert language; albeit in a rather awkward style. The resulting sets of (normalised) sentences describe a restricted subset of the expert language. This language is referred to as the* normalised information language. The conceptual schema of the universe of discourse can be regarded as a crude grammar of this latter language. This entails the following language inclusions: normalised information language $\subseteq$ expert language $\subseteq$ natural language. At present, the existing commercial ORM CASE-Tool InfoModeler is able to take sentences of the normalised format as input and convert this to a first draft of the conceptual schema ([Asy94]).

In our view, two of the remaining research challenges:

1. Build tools to aid humans in the verbalisation of facts from the universe of discourse in ORM Normalform. Such a tool could use its linguistic knowledge to propose possible normal forms. Promising results in this direction can be found in e.g. [DR91] [BR95], and [BCDT95].

2. Enhance the conceptual schema modelling language with constructs that allow for alternative, and more natural, verbalisations of facts to enrich query (and result) formulation.

This article is concerned with the second challenge. We will provide a mechanism that allows us to generate alternative and more natural verbalisations (with preferences) of a single (deep structure) fact. Normally, a conceptual query language is only able to approximate the *normalised information language in its naturalness. Using alternative verbalisations we will be able to better approximate the* expert language. This richer language is called the *information language, and its underlying grammar the* information grammar. With this language we would have:

$$\text{normalised information language} \subseteq \text{information language} \subseteq \text{expert language} \subseteq \text{natural language}$$

Simply allowing any question that can be formulated in the expert language as a query would introduce interpretation problems due to the inherent ambiguity of natural language. Therefore, our extensions are focussed on a controlled way to do this without introducing a mechanism that is prone to ambiguity. In a later section we briefly discuss the idea of building a natural language query interface that takes a query formulated in the expert language and tries to find all possible interpretations of this query in terms of queries formulated in the information language. These interpretations can then be shown to the user; leading to a natural dialogue between users and the information system.

The query language that is derived from the information language is a refinement of the existing conceptual query language LISA-D ([HPW93]. The original version of LISA-D was strictly based on the *normalised information language. The information grammar, which forms the base of the enriched LISA-D, is built from a lexicon and a set of grammar rules. Both the lexicon and grammar rules consist of application dependent and application independent parts.*

3 The Information Language

From the sample sentences which result from the analysis phase, the verbalisation and denotation of instances can be derived as well. The resulting verbose representation of instances

can be used by the information system in user dialogues. Most importantly, query formulations may benefit by improved instance denotation. Update of instances, as well as presentation of query results, can both be improved using an elegant verbalisation mechanism. In addition, sample instances may be verbalised and shown to the user as a means to validate models (see e.g. [Dal92]).

3.1 Verbalisation in ORM schemas

Before discussing the basic verbalisation functions for an ORM schema, we first introduce our running example. In figure 1, the information model of a presidential database is depicted. As ORM and its notation has now been widely published ([Hal95]) we refrain from a detailed explanation.

Any ORM information model contains two key components: types ($\mathcal{TP}$) and *roles ($\mathcal{RO}$). Types come in two important flavours:* object types and *fact types. Each fact type has associated (or can bee seen to consist of) a set of roles from $\mathcal{RO}$. The two type flavours possibly overlap as the type* Marriage *illustrates. This is an example of an* objectified fact type. All types in an ORM schema have a name, which is recorded by the function $\mathsf{ONm} : \mathcal{TP} \rightarrow \mathcal{NM}$, where $\mathcal{NM}$ is a set of names. Roles also receive a name of their own. This name is referred to as the *participation name. They are provided by the function* $\mathsf{PNm} : \mathcal{RO} \rightarrow \mathcal{NM}$. *Roles of different fact types may have identical participation names. However, two roles of the same fact type must have different participation names. Roles ($\mathcal{RO}$) may also have a* reverse participation name, $\mathsf{RNm} : \mathcal{RO} \rightarrow \mathcal{NM}$ *which verbalises the decomposition of the fact type into the components involved. As an example, consider figure 2. There we have:* $\mathcal{TP} = \{A, B, C, f\}$, *and* $\mathsf{Roles} = \{p, q, r\}$. *This is a fragment from the presidential database model, with the following names:*

$\mathsf{ONm}(A) =$ Person	$\mathsf{PNm}(p) =$ with	$\mathsf{RNm}(q) =$ obtained by
$\mathsf{ONm}(B) =$ Nr of Voters	$\mathsf{PNm}(q) =$ in	$\mathsf{RNm}(q) =$ with
$\mathsf{ONm}(C) =$ Election	$\mathsf{PNm}(r) =$ leading to	$\mathsf{RNm}(q) =$ obtained in
$\mathsf{ONm}(f) =$ Election results		

Participation names are usually omitted from schema diagrams. The actual schema diagrams usually feature the fact verbalisations that follow from the sample verbalisations. These verbalisations correspond to mix-fix predicate symbols. For instance, the verbalisation: The person with name 'Eisenhower D.D.' has the nr of voters 10000 in the Election '55' *has as underlying mix-fix predicate:* ... has ... in ... . *These mix-fix predicate symbols are captured formally by the relationship:* $\mathsf{MFix} \subseteq \mathcal{NM}^+ \times \mathcal{RO}^+ \times \{\texttt{Short}, \texttt{Possesive}\}$.

There are two forms of fact verbalisations. The `Short` *form is used for the verbalisation without possessive forms. For instance,* $\mathsf{MFix}([\mathsf{has}, \mathsf{in}], [p, q, r], \texttt{Short})$ *is a short form verbalisation for the fact type in figure 2. which with proper object type names filled in leads to:* Person has nr of voters in election. *The second form allows us to formulate facts in the* `Possesive` *format. An example would be be:* $\mathsf{MFix}([\mathsf{who\ has\ a}, \mathsf{in\ an}], [p, q, r], \texttt{Possesive})$, *which would lead to (with proper articles added):* A person who has a nr of voters in an election. *Mix-fix verbalisations for an n-ary fact type are always given as $n - 1$ (non-empty) strings with 'holes' (the '...') in which instance denotations can be substituted. So we always have:* $\mathsf{MFix}([\alpha_1, \ldots, \alpha_m], [p_1, \ldots, p_n], x) \Rightarrow m = n - 1$ *making them mix-fixed infix verbalisations. This means that a formulation like* President 'Clinton' and Person 'Hillary' are married *is not allowed, since the underlying mix-fix predicate is:* ... and ... are married *which consist of 2 holes*

and 2 strings (should be 1). The reason for this limitation is that the mix-fix predicate verbalisations are not intended as a verbalisation mechanism for instances (these are discussed in subsection 3.3), but rather as a mechanism to verbalise transitions through fact types. In particular higher (> 2) order fact types. Verbalisation of transitions through fact types are absolutely essential for the formulation of longer queries in LISA-D that span multiple fact types. An example of such a longer query is: LIST An administration which has as president a person who has a nr of voters [which is $>=$ 1000000] in an election *which spans two fact types.*

In a next step, we will try to define one single verbalisation form in a more elementary linguistic format from which the different verbalisation formats can be generated. For this, we will be looking at the use of functional grammars as a generic format ([DR91], [BR95]).

3.2 Building the Information Grammar

Using the names provided by ONm, PNm, RNm, *and* MFix *in the formulation of queries and constraints will yield expressions that are close to their original formulation in natural language. However, one should realise that we by far do not obtain the richness of a natural language. This apparent disadvantage, however, comes hand in hand with the advantage that the inherent ambiguity of natural language is absent.*

Designers of conceptual query languages necessarily have to perform a delicate balance act between on the one side the requirement to define a mathematically based and unambiguous language, and on the other side the need to define a language that is close to natural language. The result is a language in which expressions have a clear intuitive meaning as the expressions are close to natural language, but which handles 'sluggish' when formulating expressions. Interpretation of the word 'sluggish' depends on one's background. From a natural language point of view, the resulting language is certainly not flexible and elegant. When looking from an SQL point of view though, the resulting language is highly flexible, orthogonal, and closer to one's intuition. Finally, by adding tool support helping users in the query formulation process, the 'sluggishness' of the language should be dramatically lessened.

The basis for the information language is the lexicon, assigning a meaning to the names given to concepts in the information structure. The meaning of names is administered by the relation Lexicon $\subseteq \mathcal{NM} \times \mathcal{PE}$ *where $\mathcal{PE}$ are so-called path expressions. The path expression language is defined in full detail in [HPW93], and basically corresponds to a path oriented version of the relational algebra that utilises the graph nature of information structures. This language is used as the semantic base of LISA-D.*

The lexicon is filled from three sources. The names of the concepts in the information structure are the first source:

1. *Names of types themselves:* ONm(x). *For example:* president.
2. *Type names preceded by an appropriate article:* Article(x, k) ONm(x). *Examples are:* the marriage *and* a person. *The articles, like:* the, an, a, *of a type is presumed to be provided by the function:* Article : $\mathcal{TP} \times \{\texttt{Undetermined}, \texttt{Determined}\} \rightarrow \mathcal{NM}$ *Though articles can be specified manually, linguistic tools to automatically generate articles do exist.*
3. *Fact participation names:* PNm(x). *For instance:* is spouse in.
4. *Reverse fact participation names:* RNm(x). *For instance:* with spouse.

The second source are generic keywords for standard constructs in an information structure, like bridge types, collection types, sequence types, etc. In this article we will not discuss these keywords in detail. Finally, the third source of keywords are the constants, like 'Ford G.R.' *To build the actual language from the lexicon, a number of construction mechanisms is used. The most important one is concatenation, which comes in two flavours. Firstly, we can simply juxtapose words from the lexicon:* Election leading to election results obtained by person*. The second flavour of concatenation allows us to use the mix-fix predicate verbalisations. On each position of a mix-fix predicate an existing sentence may be inserted, thus creating more complex sentences. To avoid ambiguities, we sometimes need to add brackets. An example of such a case is:* An administration which has as president a person who has a nr of voters [WHICH IS $>=$ 1000000] in an election*, which is built using the mix-fix predicate verbalisation* ... has ... in ... *for the election results fact type.*

Besides concatenation, a number of other construction mechanisms exist. For example, operations like union and intersection, predicates, and comprehension schemes. A detailed discussion of these language constructs can be found in [HPW93]. To allow for the introduction of more compact and convenient verbalisations, LISA-D also supports a macro mechanism. An interesting aspect of this mechanism is that it allows for the specification of arbritrary fix-point queries ([HPW97]).

3.3 Denotation of instances

To support the denotation of instances, in particular instances of complex object types, it is assumed that each object type has associated a number of verbalisation rules. We have designed a scalable verbalisation mechanism that allows us to define elegant and situation specific denotations of instances. Verbalisation rules specify how object instances are represented in the universe of discourse. As this is the preferred way to denote instances as used by the parties involved in the universe of discourse, it makes sense to ensure that in the communication with the information system these denotation styles can be used as well.

Element verbalisations usually omit explicit verbalisation of typing information. For example, we would use 'J.F. Kennedy' *in the presidential database example if only referring to* 'J.F. Kennedy' *as a person. When referring to this person as a president, this would become* president 'J.F. Kennedy'*. This way of treating typing information, and using it only when needed, is a convention used in natural language that keeps sentences readable. However, when we integrate the element verbalisations in the LISA-D language, special care must be taken not to introduce ambiguities this way. In some situations explicit typing must simply be added.*

Verbalisation rules are treated as meta-rules in the sense of two-level grammars (see e.g. [WMP+76]). They are instantiated by a population of the information structure. As a result, with any population a concrete verbalisation grammar can be associated. For our running example, we have the following default verbalisation with regards to the fragment dealing with people's addresses:

$$\begin{aligned}
\langle \text{Community} : x \rangle &\rightarrow_1 \langle \text{Community name} : \text{Community name of Community } x \rangle \\
\langle \text{Street} : x \rangle &\rightarrow_1 \langle \text{Street name} : \text{Street name of Street } x \rangle , \langle \text{Community} : \text{Community contains Street } x \rangle \\
\langle \text{Address} : x \rangle &\rightarrow_1 \langle \text{House Nr} : \text{House Nr of Address } x \rangle \langle \text{Street} : \text{Street of Address } x \rangle
\end{aligned}$$

The first rule states that an instance x of type Community *is verbalised by a* Community name*. This community name is determined by* Community name of Community x*, i.e. the community name of*

community x. To denote a street, the house number and street name are denoted first. This is followed by a ',' and the community name. This leads to a denotation of addresses in the following format: 1 Toernooiveld, Nijmegen. In some countries, this is not the accepted style to write addresses. For example, in some countries the preferred style to write this address is: Toernooiveld 1, Nijmegen. What makes this case special is that while a street is identified by a street name (Toernooiveld) and the community name (Nijmegen), the house number needs to be placed in between. To obtain this kind of verbalisation we need to introduce two additional (partial) verbalisation rules for Street.

$$\begin{aligned} \langle \text{Street} : x \rangle &\rightarrow_2 \langle \text{Street name} : \text{Street name of Street } x \rangle \\ \langle \text{Street} : x \rangle &\rightarrow_3 \langle \text{Community} : \text{Community contains Street } x \rangle \end{aligned}$$

Note that the subscripts 1, 2, and 3 of the right arrows ($\rightarrow$) reflect the preference on the verbalisation rules. However, there is more to this numbering than meets the eye. This is explained in more detail in [HPW97]. For this article it suffices to simply presume that the numbers provide the preference, where 1 has a higher preference than 3.

If we also change the verbalisation of Address *to:*

$$\begin{aligned} \langle \text{Address} : x \rangle &\rightarrow_1 \langle \text{Street} : \text{Street contains Address } x \parallel 2 \rangle \langle \text{House nr} : \text{House nr of Address } x \rangle , \\ &\quad \langle \text{Street} : \text{Street contains Address } x \parallel 3 \rangle \\ &\rightarrow_2 \langle \text{House nr} : \text{House nr of Address } x \rangle \langle \text{Street} : \text{Street of Address } x \rangle \end{aligned}$$

the desired verbalisation style is obtained. Using $\langle t : P \parallel l \rangle$ we express the preference to use verbalisation rule l for type t in the context of the given rule P.

Fact types also have associated verbalisation rules. For fact types it is also very useful to introduce partial verbalisations, which only verbalise part of a fact instance; focusing on a number of aspects only. For example, the participation of a person in an election, irrespective of the numbers of voters, may be verbalised as:

$$\begin{aligned} \langle \text{Election results} : x \rangle \rightarrow_2 \ & \text{the participation of } \langle \text{Person} : \text{has Election results } x \rangle \\ & \text{in } \langle \text{Election} : \text{with Election results } x \rangle \end{aligned}$$

In the inheritance hierarchy, the default is that verbalisations are inherited downward. However, the inherited verbalisations may be overriden. In case of the running example, the verbalisation of instances of President *differs from the verbalisation of instances of* Person. *The following example shows how this can be handled.*

$$\begin{aligned} \langle \text{Person} : x \rangle &\rightarrow_1 \langle \text{Politician} : x \rangle \\ \langle \text{Politician} : x \rangle &\rightarrow_1 \langle \text{President} : x \rangle \\ &\rightarrow_2 \text{politician } \langle \text{Person} : x \rangle \\ \langle \text{President} : x \rangle &\rightarrow_1 \text{president } \langle \text{Person} : x \rangle \end{aligned}$$

4 Computer Supported Query Formulation

In this section, we study as an example application how an integrated workbench could be built that tries to combine the formality and completeness of LISA-D with the freedom of

natural language querying. A query workbench should contain a set of integrated and complementary tools, each of which supports a part in the query formulation process. We propose such a workbench to have at least the following tools: Query by Navigation, *Natural Language Querying, and* Query by Construction. In the remainder of this section we briefly highlight each of these tools.

Query by navigation is a mechanism that has been succesfully used in the context of information retrieval ([AAC+89], [Bru90], [BBB91]). Its application in the context of query formulation on structured databases has been reported before in e.g. [HPW95]. In figure 3 a fragment of a query by navigation session is shown. The central idea of query by navigation is to let users navigate through the graph spanned by the information model, meanwhile familiarising themselves with the information available, and gradually formulating part of their query. In figure 3, the user has arrived at the president who is involved in a marriage, and can now refine the focus to e.g. the president who is involved in a marriage with as spouse a person, or further enlarge their focus to president.

A language like LISA-D is clearly better suited for non expert users to specify their own queries, then a language like SQL. However, due to its unavoidable formality, LISA-D as such may still be regarded as too limiting for end users. Therefore, it makes sense to also include a natural language front end. Given a query in full natural language, this query could be interpreted as a path expression using the verbalisations specified with the conceptual schema.

In figure 4 an example natural language query is given. This query is ambiguous in that it is not clear to which person the who is the vice president of an administration refers to. The system would therefore be able to interpret this query in at least two ways. By translating each of these interpretations to a path expression (the semantic domain of LISA-D), and then reverbalising these path expressions, we could built a feedback system that enables the system to have a natural dialogue with users. In figure 5 the system shows the different interpretations to the user.

The query by navigation tool and the natural language front end together still do not allow users to build queries with all kinds of complex operators. It is clear that query by navigation as such does not support this. Natural language querying will support this only to some extent. Formulating a query requiring a fix-point for its evaluation can be very tedious when using natural language. This is not a flaw of query by navigation or natural language querying. The aim of these two tools was respectively to help users explore the stored information, and to allow them to formulate queries in their own words.

To support the formulation of arbitrarily complex queries, we therefore need a tool that basically provides a syntax directed editor. An example of this is shown in figure 6. This *Query by Construction tool basically provides a query workbench. Query 'snippets' that either result from a query by navigation session, or natural language query, can be dragged onto the query workbench where they can be combined into complex queries using an array of operations. The power of such a query workbench is obviously the combination of tools.*

5 Conclusions & Discussion

Focus in this paper has been on grammatical aspects of conceptual data modelling, where we did not yet dare venture into the area of linguistics. The latter step will be part of future

research in an attempt to further improve the naturalness of LISA-D and instance verbalisations. We have shown how the use of verbalisations allows information systems to communicate with users in a language that more closely resembles their language. Query results can be presented in a more user-friendly format. The verbalisation mechanisms described in this paper are very liberal.

Future research now focuses on three issues. The first issue is the automatic support of the first phase, i.e. the automatic processing and refinement of the sample verbalisations. Secondly, we intend to specify verbalisation information in a more generic format to allow more flexible use in LISA-D. This generic format would have to correspond to what in [All95] is called the logical form. In [All95] a distinction is made between an actual natural language sentence, its logical form, and the final representation. For LISA-D the final representation would be the underlying path expression language. Functional grammars would be a possible candidate to introduce this logical form.

Finally, we are also studying how verbalisations can help in advertising databases in a networked environment; characterising the contents of a database.

Acknowledgements

The work reported in this paper has been funded in part by the Cooperative Research Centres Program through the Department of the Prime Minister and Cabinet of Australia.

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.

[All95] J. Allen. *Natural Language Understanding*. Benjamin Cummings, Redwood City, California, 2nd edition, 1995.

[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.

[BCDT95] E. Buchholz, H. Cyriaks, H. Düsterhöft, A.and Mehlan, and B. Thalheim. Applying a Natural Language Dialogue Tool for Designing Databases. In *Proceedings of the First International Workshop on Applications of Natural Language to Databases (NLDB'95)*, pages 119–133, Versailles, France, June 1995.

[BR95] J.F.M. Burg and R.P. van de Riet. COLOR-X: Object Modeling profits from Linguistics. In *Proceedings of the KB&KS'95, the Second International Conference on Building and Sharing of Very Large-Scale Knowledge Bases*, Enschede, The Netherlands, 1995.

[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.

[Dal92] H. Dalianis. A method for validating a conceptual model by natural language discourse generation. 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 425–444, Manchester, United Kingdom, 1992. Springer-Verlag.

[DR91] F.P.M. Dignum and R.P. van de Riet. Knowledge base modeling based on linguistics and founded in logic. *Data & Knowledge Engineering*, 7:1–34, 1991.

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

[HPW95] 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.

[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.

[Kri94] G. Kristen. *Object Orientation – The KISS Method, From Information Architecture to Information System*. Addison-Wesley, Reading, Massachusetts, USA, 1994. ISBN 0201422999

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

[RBP+91] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, and W. Lorenson. *Object-Oriented Modeling and Design*. Prentice-Hall, Englewood Cliffs, New Jersey, 1991.

[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.

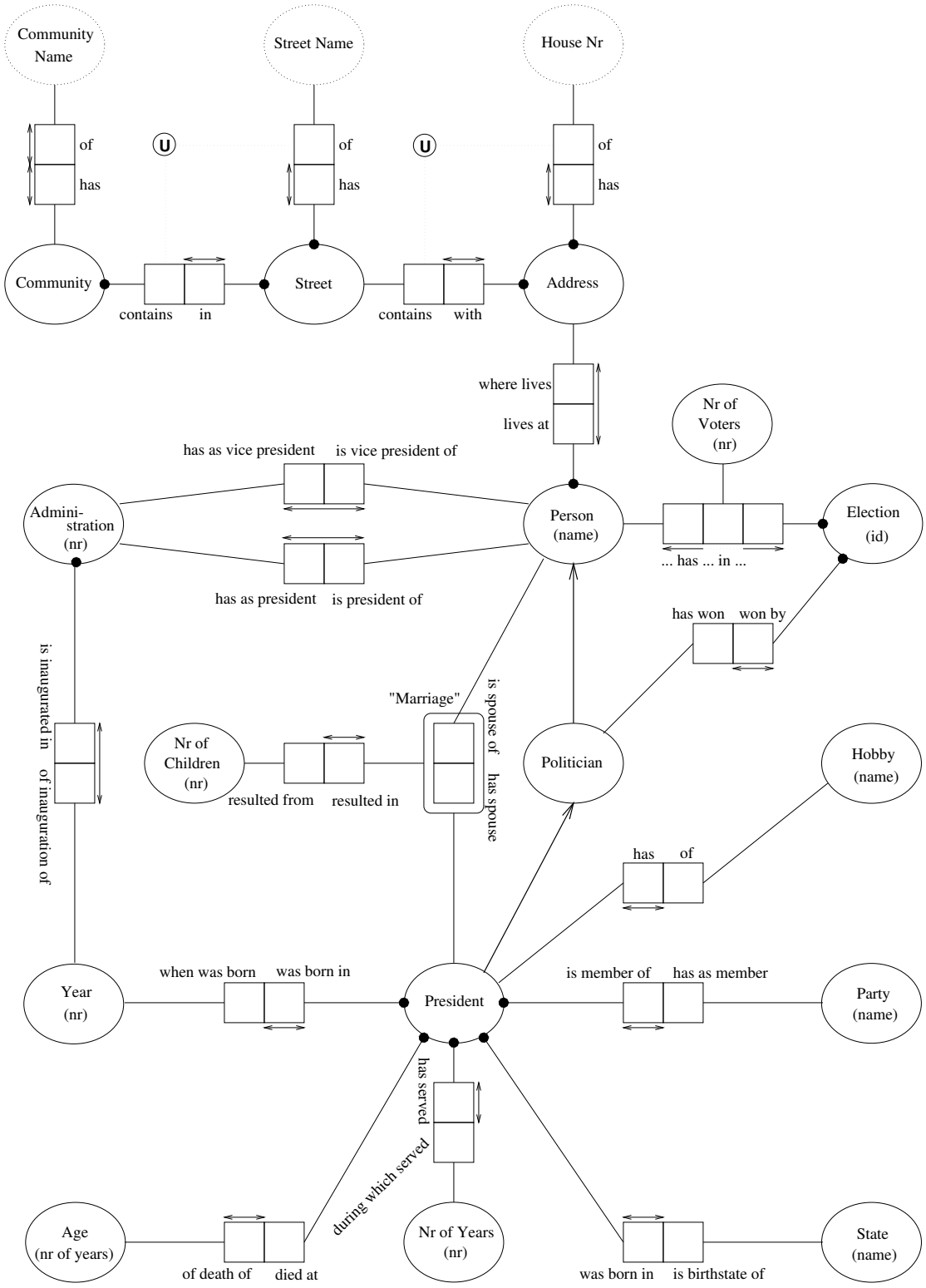

Figure 1: The presidential database information model

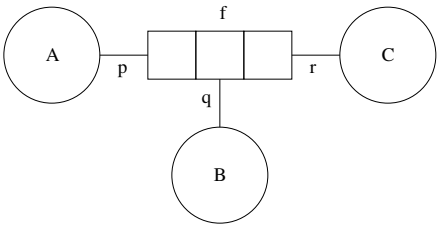

Figure 2: Example model before naming

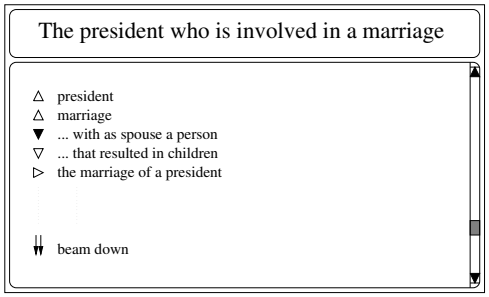

Figure 3: A Query by Navigation session

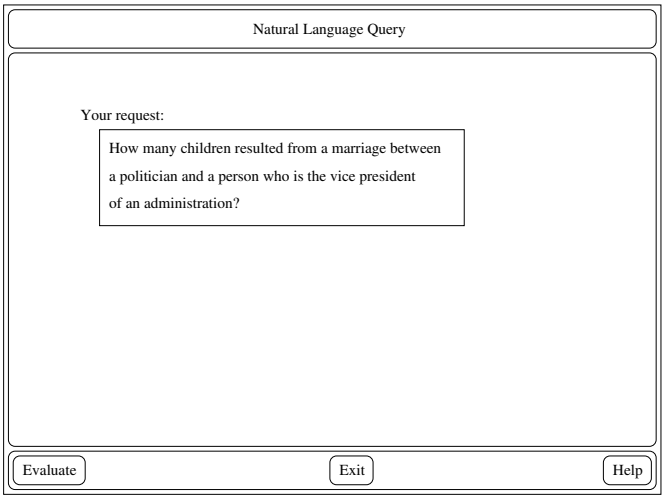

Figure 4: A natural language query

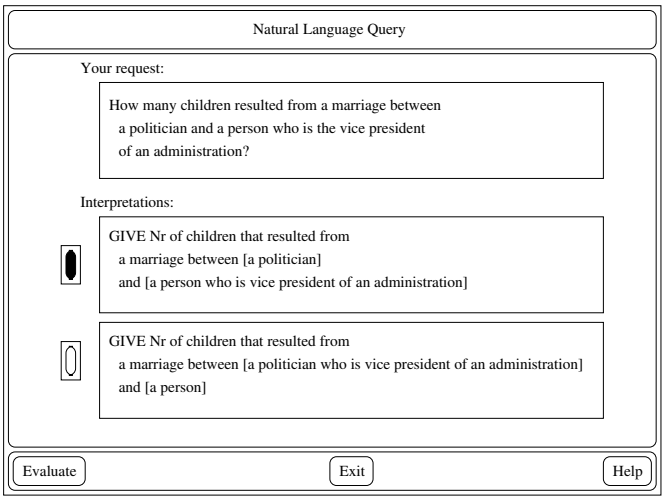

Figure 5: Resolving ambiguities by re-verbalising interpretations

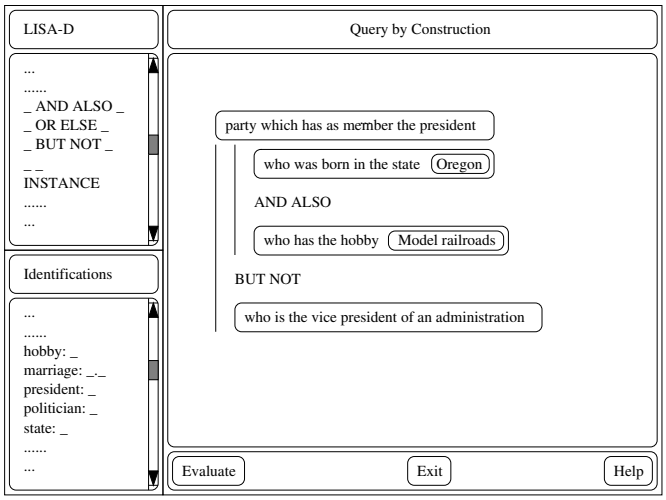

Figure 6: Example query by construction session