Pasta-3's Graphical Query Language: Direct Manipulation, Cooperative Queries, Full Expressive Power

Michel Kuntz and Rainer Melchert

ECRC, Arabellastrasse 17, 8000 Munich 81, West Germany

Abstract

Graphical Direct Manipulation (GrDM) query languages are characterized and positioned with respect to traditional textual query languages. After a brief survey of the current state of the art, the Pasta-3 GrDM query language is presented, with emphasis on three features that make significant new contributions to this field: (1) its GrDM basis (query editing through clicking and dragging of an iconic representation of the query expression), (2) its cooperative environment (handy values, automatic path completion, edit-and-reevaluate loop), and (3) its expressive power (arbitrarily complex conditions, recursive queries, logical variables and quantification, subqueries, mixing in Prolog code). Several examples taken directly from the nearly complete implementation are discussed in detail.

1. Introduction

Query languages as a data base research topic need no introduction -- nor any justification. Until recently query languages were linear textual languages: of the three usual levels of linguistic analysis -- semantic, syntactic, lexical -- only the first one was considered important. The other two were routine: syntax was relegated to a BNF grammar in an appendix, and the lexical level was merely a matter of character strings, not worth mentioning as such. Besides this, language design was the main research activity, since implementation of such languages posed few problems (optimisation excepted) thanks to the maturity of compilation techniques. Thirdly, using such languages was no matter for discussion: there was little to discuss -- you simply typed in statements.

This one-sided approach was inevitable as long as a small number of professional programmers were the only target user group and the only I/O devices widely available were alpha-numeric terminals and keyboards. Today this is no longer the case, and there is good reason to ask if the triple imbalance just described should not be corrected.

A shift towards a better balance has in fact already begun with the emergence of **graphical** query languages. Work in this specific area has been motivated by two factors:

- the realisation that query languages are not only pure mathematical objects but also part of interfaces intended for actual practical use, and
- the technical and economic availability of hardware (high resolution, bit mapped screens and pointing devices such as mice) and software (window managers and graphics packages) supporting graphical interaction.

The graphical Direct Manipulation (GrDM) interaction paradigm[1] usually involves selection of items from pop-up menus and use of icons that can be clicked on and dragged: visible graphical objects both represent the current state of the interface and can be acted upon by "direct" physical actions: moving the tracking symbol over the screen and pressing mouse buttons. GrDM was first used commercially in the Xerox Star and Apple Lisa systems. It has already contributed to the success of software such as spreadsheets, WYSIWIG text editors, CAD systems, etc.

Graphical querying facilities have become an active research area since the mid 1980s. However the old tendency to privilege semantics and design and to ignore actual use is still alive and well: quite recent work in this area [Campbell 87, Catarci 88, Cruz 87, Dart 88] focusses on abstract definitions of new languages and says very little about how they actually look on the screen or how someone would go about using them. On the other hand, some research [Bryce 86, Goldman 85, Kim 88] has helped to restore the balance and place emphasis on the previously neglected aspects of graphical querying. The work reported in this paper, Pasta-3, is a further step in this direction.

The basic approach adopted here is not to define a new formal language but to provide a sophisticated GrDM editing capability for the original formal linguistic objects (already well-defined semantically in the back-end DBMS: see Section 2.2). Put another way, it is the same semantics associated with a GrDM (rather than a textual) syntax.

The Pasta-3 interface includes GrDM capabilities for **all** types of DBMS manipulation activities.[2] However, the purpose of this paper is only to present certain important features of the Pasta-3 **query language**:

[1] [Shnei 83] and [Hutch 86] are just two of the many publications that provide more complete discussions of GrDM.

[2] See [Kuntz 89a, Kuntz 89b, Kuntz 89c] for the other aspects of Pasta-3.

- its GrDM basis: query editing through clicking and dragging of an iconic representation of the query expression (Section 3),
- its cooperative environment: handy values, automatic path completion, edit-and-reevaluate loop (Section 4),
- its expressive power: arbitrarily complex conditions, recursive queries, logical variables and quantification, subqueries, mixing in Prolog code (Section 5).

All three aspects make significant new contributions to the area of GrDM query facilities.

But before these main topics can be properly explained, three items of background information need to be supplied. Since Pasta-3 is not alone in its field, a short survey of related work to date will help to clarify relative positions (2.1). A query language is difficult to understand without reference to the DBMS it is meant to work with -- the KB2 System in this case (2.2). Finally, Pasta-3's query language is part of a complete, integrated interface and can only be understood in this context (2.3).

A note on the implementation and the conclusion make up the last section (6).

2. Background

2.1. State of the Art

Non application domain specific research[3]has produced at least 20 published results: [Bryce 86], [Campb 87], [Catarci 88], [Cruz 87], [Dart 88], [Du 88], [Elmas 85], [Gimn 88], [Goldm 85], [Kim 88], [King 84], [McDon 75], [Melch 87], [Motro 88], [Teskey 84], [Udaga 82], [Urspr 83], [Wong 82], [Wu 86], [Zhang 83].

Analysing them individually and in detail would be the subject matter for another whole paper.[4]Only a few general observations can be made here.

The term "graphical" is **ambiguous**. One sense is abstract and general: pictorial or diagrammatic (as opposed to textual). Another is concrete and restricted: pertaining to raster operations on bitmaps. A third is mathematical: pertaining to graphs. The literature applies an inclusive OR over all three meanings when speaking of query facilities. All results mentioned above are graphical in the first sense, but in the second and third, some are and some are not.

The **major contribution** of graphical querying has been to reduce the cognitive load on users in several ways: lack of typing skill is no longer a significant handicap, formal syntactical conventions are largely replaced by intuitive GrDM operations, and active memorized knowledge of the DB is rarely required since schema components are already visible on the screen and need only be passively recognized.

All the results cited above are **combinations of graphics and text** -- rather than purely graphical systems. In some cases [Catarci 88, Dart 88, Elmasri 85, Ursprung 83, Wong 82, Wu 86, Zhang 83] graphics are used essentially for schema visualisation, and query formulation relies mainly on form-filling or typed-in expressions.

Some graphical query facilities [Goldman 85, King 84, Teskey 84, Wong 82, Zhang 83] do not rely on any explicit graphical **language** at all. They are defined procedurally, in terms of the operations needed to input a query -- in this case operations other than typing characters. There is no well-defined area of the screen the appearance of which completely expresses the query. One cannot simply "read" some configuration of shapes and text to know what was indeed the intent of the preceding sequence of operations.

On the other hand, some query facilities [Campbell 87, Catarci 88, Cruz 87, Dart 88, Udagawa 82] offer only a naked language, with no accompanying **environment** (in the usual sense: a well-integrated set of facilities that support / enhance actual use of the (query) language). Experience in software engineering has demonstrated the importance of good environments for programming languages -- the same holds true for query languages.

There tends to be a **trade-off** between expressive power and ease of implementation: powerful graphical languages have sometimes not been implemented, and many effectively running graphical query facilities support only very simple queries.

This situation indicates that the three main points where progress needs to be made involve:

1. extending **expressive power** to handle complex queries, while
2. taking better advantage of the **GrDM** style of querying, and
3. combining the benefits of a **language and an environment.**

2.2. The KB2 System

Pasta-3 is an interface to KB2 [Walla 86], a knowledge base system embedded in Prolog -- or more exactly -- in EDUCE [Bocca 86], a logic programming data base system that supports both a loose coupling and a tight integration of Prolog and a relational DBMS. KB2 uses the Entity / Relationship data model, extended with inheritance and deductive rules. It was developed at ECRC by Mark Wallace and other members of the Knowledge Base Group.

KB2 provides facilities for structured information storage on disk. Queries and updates expressed in terms of

[3]Application domain specific work such as [Herot 80, Chen 87, Egenh 88] is not considered here.

[4]and extending the field of comparison to include graphical (or "visual") programming (cf. [Raeder 85], [Myers 86], [Reiser 88]) and graphical interfaces for other logical constructs such as inference rules (e.g. [Lewis 83], [Poltr 86]) is also regretfully excluded

its extended E-R model are mapped down to their relational equivalents. Both schema and data can be modified at any time. However, all updates are checked for consistency with a set of integrity constraints. The information represented with KB2 includes both basic facts and deduction rules. Enforcing the integrity constraints just mentioned ensures that all information -- what is deducible as well as data stored as such -- is always consistent.

KB2 extends traditional data bases in two main directions:

- it supports **general** information through deduction rules and integrity constraints, and
- it allows applications to have a completely **dynamic** knowledge structure.

This dynamic character means that, throughout the life of an application, new Entities and Relationships can be created, the inheritance lattices extended, and properties of E-R items added or deleted.

KB2 supports **complex logical queries**, including quantification, limited recursion, and mathematical functions, expressed as predicate calculus formulae, which are mapped onto Prolog goals and clauses. The fact that KB2 offers its users a manipulation language of this power has important implications for Pasta-3: if its GrDM query language is not equally expressive, end-users would be reluctant to accept a trade-off of more ease of use against loss of expressiveness.

2.3. Overview of Pasta-3

This section provides general background information on Pasta-3 and gives some brief notes on its top-level architecture.

Figure 1 shows a partial view of one possible screen with several different types of windows.

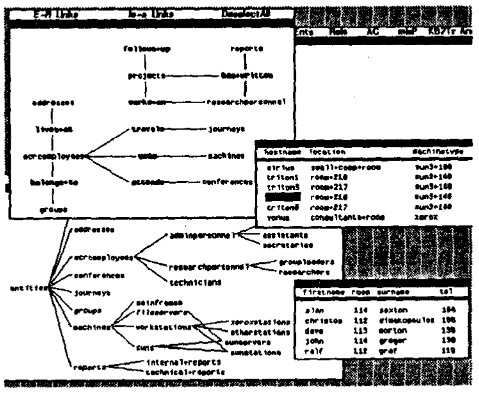


Figure 1: Partial view of screen (one possible layout)

On the left are two graphical schema manipulation windows (E-R schema and Entity inheritance lattice). They provide both schema creation and schema browsing functionality. Two tabular data display windows are visible towards the lower right: they could be the result of browsing or querying. In the upper right quadrant one can see a (still empty) querying workspace.

Figure 2 shows Pasta-3's intermediate position between the workstation and the back-end DBMS, KB2.

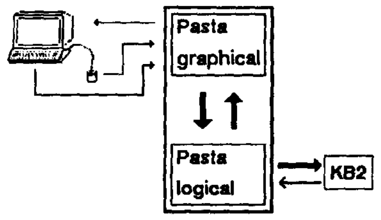


Figure 2: Top-level architecture of the Pasta-3 interface

Pasta-3 is composed of two main parts: **Pasta Logical** and **Pasta Graphical**. The latter is "nearest" to the user and is directly responsible for dealing with all I/O that passes through the interface. In the input direction it translates user activity manifested by means of mouse and keyboard into expressions of the language used as an internal communication protocol connecting to the other main component, Pasta Logical. In the output direction, Pasta Graphical takes character-strings containing information generated at lower levels of the overall system and produces the appropriate visual representation on the screen. The jobs of Pasta Logical are to construct KB2 expressions from the protocol language commands, to invoke KB2 processing, and to collect arbitrary numbers of variable instantiations returned from KB2 into a single result list, suitable for transmission up to Pasta Graphical.

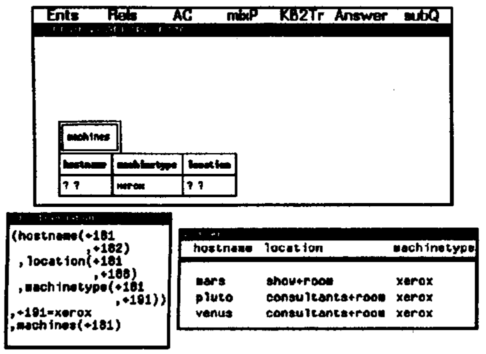
3. Basic GrDM Characteristics

Pasta-3's querying capability has been designed to support all queries that are expressible in KB2's linear language (which is logical rather than navigational). The query specification process takes place in an instance of a dedicated window type (a "workspace") and results in exactly one expression in Pasta-3's two-dimensional language. The steps composing this process can be carried out in whatever order the user prefers.

For presentation purposes, it is helpful to distinguish the core language elements from relatively sophisticated aspects. The basic representations and operations used in querying are illustrated with a rudimentary example having only one Entity in the workspace and a compound selection consisting of one conjunction of selections all with the comparator "=". At this point three of Pasta-3's cooperative features can be explained (Section 4). Next (in 5.1) certain more advanced aspects will be discussed together from the starting point of a query involving several Entities and Relationships in the workspace, possible use of aggregate functions, and a selection consisting of an arbitrarily complex tree of compound selections (AND, OR, NOT) containing selections using various comparators. Finally (5.2) we move on to other powerful options illustrated by four final examples.

3.1. Graphical Representation

The first example query is "name and location of the Xeroxes ?" Fig. 3 below shows three related windows: the large one is the query workspace containing an elementary query expression, the small one on the left shows the query translated into the KB2 language, and the third one is a data display window. This strictly tabular way of expressing elementary queries originated with QBE [Zloof 77] and was previously adopted for HIQUEL [Urspr 83], GLAD [Wu 86], and Pasta-II [Melch 87].



```
(hostname(+181
          ,+182)
 ,location(+181
          ,+188)
 ,machinetype(+181
          ,+191))
,+191=xerox
,machines(+181)
```

```
hostname  location           machinetype

mars      show+room          xerox
pluto     consultants+room   xerox
venus     consultants+room   xerox
```

Figure 3: Query workspace, KB2 translation, and answer

The double-bordered rectangle in the lower-left corner of the workspace represents the Entity "machines". Three of its properties ("hostname", "machinetype", and "location") are already available for further work. The "??" projection symbol in the boxes below "hostname" and "location" indicates that results (i.e. data) for these properties are to be displayed. The string "xerox" boxed below "machinetype" represents a selection, the "=" comparator being implicit here by convention (as the conjunction would be if there were several selections).

Display of the linear syntax translation is normally entirely optional: it is necessary only in conjunction with the "Mix in Prolog" capability (see 5.2 below).

3.2. Query Building Operations

How did this very simple query expression come into existence ?

Once a new workspace has been created (via the Pasta-3 main menu), the first step in using it is to indicate which Entities and/or Relationships the query involves. To do this, users have two possibilities: if they have been browsing, they can copy into the workspace E-R items already displayed in other windows (cf. Fig. 1); if they have opted for querying directly, the attached menu at the top of the query workspace window provides the required functionality: two of its items are "Ents" and "Rels". Clicking on these pops up lists of Entities and Relationships from which the ones of interest can be chosen. Either possibility produces the same effects: an icon representing the E-R item in question appears in the workspace and the corresponding icon in the schema graph windows (cf. Fig. 1) is highlighted. In the example, the user has chosen the "machines" Entity.

Clicking the "Properties" item in the pop-up menu for "machines" (leftmost in Fig. 4) displays the list of this Entity's properties from which the user can choose one or more for further work. In Figure 3 above, the user has already chosen three properties: "hostname", "machinetype", and "location".

To indicate that they want results for the "hostname" property to be displayed, users pop up the command menu for properties (second in Fig. 4) by clicking on "hostname" in the workspace in Fig. 3 and choose the "Display" item. This immediately causes the display of the "??" projection symbol in the box below "hostname". They do the same for "location".

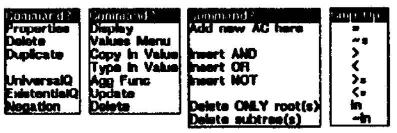


Figure 4: Menus: ER items, properties, conditions, comps

Specifying the selection will complete the query. Two of the selection's three parts -- the property name and the comparator -- are already available: the former is visible on the screen and the latter is implicit (in this case -- for other comparators, see below). Specification of the value "xerox" can be done in any of three possible ways, explained in Section 4.1 below.

Clicking on the "Answer" option in the attached menu of the workspace window invokes query evaluation and displays the answer in a separate data display window.

4. Cooperative Querying

GrDM contributes to ease of use in a non-application-specific way. There are also specific ways in which a query language environment can make life easier for its users. We have grouped three of them together in this section.

4.1. Handy Values

Most queries -- no matter what the language -- contain constant values as part of the specification of the selection. Indicating such values is a very frequent operation in query formulation and one that is particularly error-prone when it involves typing and memory recall. Making it easier would be a big gain for any query interface. Pasta-3 provides three ways to do it.

The easiest one to use is choosing the desired value from the menu displayed in response to a click on the "Values menu" item in the properties pop-up menu (Fig. 4). This item is present in the menu only for those properties that have no more than 15 different possible values.[5]This is shown in Fig. 5.

[5]an empirical and somewhat arbitrary limit: users have no difficulty in interacting with a menu of 15 items in this context, but a larger or smaller number of values could be used.

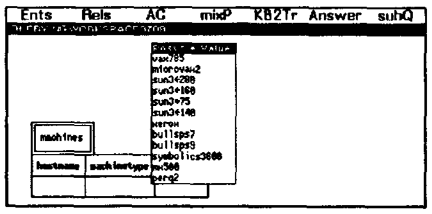


Figure 5: Choosing a value from a menu of possibilities

This method of indicating a value contributes to cooperative querying by anticipating and avoiding some of the problems addressed by work on "cooperative answers" [Kaplan 82, Wahls 85]. Instead of having to compute an explanation of the result "No data found" caused by the use of an unknown value in a restriction, the system encourages (but does not force) users **not to enter** any such values, thus eliminating the empty answer and the need to explain it.[6]

Use of such value menus has its price in interface complexity and computational cost. Cost is mostly incurred once when Pasta-3 is first set-up for a given DB: at that time all the data has to be counted, values collected for properties with less than 15 (or n) different values, and menus built. With the toy DB we have been using for demos this takes about five minutes. With very large DBs the cost might be prohibitive. Interface complexity derives from the fact that whenever updates are performed, Pasta-3 must build, destroy, and/or modify menus dynamically (but this entails no user-perceptible performance overhead).

A more serious problem will arise when multiple users access the same DB: in this case updates occur without the interface knowing of them. This problem is more general than just handy values menus -- it raises fundamental questions about the relationship between interfaces and backends, since the only solution available seems to be modifying the DBMS itself so that: it remembers updates that happened since the last login of user X and informs X's current session with Pasta-3 of these changes, and it does mutual "tipping off" of updates among concurrently running sessions.

The second possibility is to use the "Copy In value" command. This can be invoked when values are already on display elsewhere on the screen in data display tables (as a result of data browsing or as the answer to some previous query) . In this case also, user typing errors are obviously excluded. However, one cannot guarantee that empty answers will be avoided, since there are no constraints on which values can be copied from where to where.

The third possibility is the traditional one: entering a character string from the keyboard.

4.2. Automatic Path Completion

Many complex queries involve several Entities and Relationships. The most frequent case that we have encountered (and the simplest one) concerns two Entities -- that users select correctly -- and one Relationship -- that users forget to select -- even though their intention was that the two Entities should be considered connected. In less simple cases the missing connection can include several Entities and Relationships and be ambiguous (i.e. be a subgraph instead of a path). The difficulties involved here have already been studied [Maier 83, Zhang 83, Pahwa 85]. Traditionally users have a semantic problem to solve in navigating the right connection and syntactic one in building the linear textual expression that corresponds to the chosen connection. Since getting it wrong is easy, helping users to get it right is again an important advantage for any interface.

Pasta-3 is cooperative in this respect thanks to its ability to compute missing connections automatically: if the workspace contains E-R items that are connectable but not yet directly connected, and evaluation is invoked, the system will find the missing items and either offer to add them in (if exactly one connecting path exists) or propose a choice (if ambiguous possibilities exist). Doing these computations is one of the major functions of the logical subsystem of Pasta-3.

4.3. Edit-and-Reevaluate Loop

As soon as a query has been built and evaluated[7], results are displayed. On seeing them, users often realize that the query did not in fact express what they had wanted. In other cases, the query was indeed faithful, but they want to see results for a slightly different query and compare them to the previous ones. Any interface that does not provide convenient means to support these two querying behaviours[8]is not cooperative. Pasta-3 does and is.

The query expression in a workspace can be modified, using the same GrDM techniques as in the building phase, and reevaluated, with the new answer data appearing in another display window, as many times as the end user wishes. Support for such an "edit-and-reuse" way of working constitutes an essential factor contributing to good productivity.

5. Extended Expressiveness

As so far presented, the Pasta-3 query language is very convenient to use but has very little expressive power. One must retain this ease of use (i.e. not revert to typing in statements of a textual language) while greatly extending expressiveness. This section gives some idea of how Pasta-3 solves this problem. However space constraints

[6]except possibly in queries incorporating open (i.e. set) subqueries

[7]KB2 as such supports lazy evaluation, but for the time being Pasta-3 only uses immediate evaluation.

[8]also termed "retrieval by reformulation" in other interfaces (e.g. RABBIT [Willi 84] and KARMA [Bose 86])

make it impossible to include in the discussion of examples explanations of the processing done by the logical subsystem of Pasta-3 -- it must nonetheless be pointed out that much of the overall power of the interface derives from the logical subsystem's flexible internal representation of queries and the sophisticated transformations it carries out.

5.1. Complex Selections

The second example query can be paraphrased in English as: "How many researchers between the ages of 20 and 30 not in rooms 120, 220, or 320 belong to the HCI group ?" Fig. 6 shows the Pasta-3 formulation.

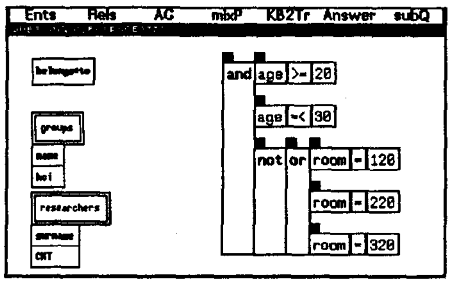


Figure 6: A second example query expression

Aside from the introduction of the count aggregation function (CNT), this example differs from the first one mainly in two ways:

- several E-R items are involved, and
- the selection is complex.

Users can bring Entities and Relationships into the workspace in the ways described for the previous example. Building a complex selection expression proceeds as follows. The user starts by constructing a single "atomic condition". This process begins with clicking on the "AC" item in the attached menu and leads to the state where an empty atomic condition icon has been created at the right of the workspace, ready to be instantiated.

To express that the "age" property is to be the left operand of this condition, the user brings up a menu of the properties of E-R items currently in the workspace and chooses "age". To specify the desired comparator, the user clicks in the middle box of the condition, pops up the comparator menu (rightmost in Fig.4), and opts for ">=". Specifying the right operand can in general be done in the three ways mentioned in Section 4.1 (menu selection, copy, type in).

The user now clicks on "AC" again to create a second empty condition box and fills it in analogously to the first. To add the conjunction connecting these two conditions, the user selects all the conditions to be connected by clicking in the small black squares at the top left corner of each condition. The "AND" connector can be obtained by clicking on the black square of the last condition to be chosen, getting the third pop-up menu in Fig. 4 and selecting "Insert AND". An AND logical operator icon is then put in place automatically by Pasta-3.

It is important to point out -- but impossible to show on paper -- that the structure of the condition tree can be modified at will by **dragging** the icons composing it. If an atomic condition or a logical operator icon is clicked, it (plus its entire subtree in the case of a logical operator) can be pulled away from its location in the tree and dropped elsewhere. Moving it away detaches it. Dropping it on a logical operator attaches it as the last operand of that operator. Dropping it in the background simply leaves it unattached at top level. In all cases, the graphical condition tree is automatically reshaped and redisplayed to take into account the changes in its logical structure. This GrDM capability provides an easy way to build up very complex logical conditions and thus removes one of the major difficulties encountered in using traditional, linear query languages: users get lost in the syntax of complex expressions before they manage to formulate the condition that they had in mind.

The user builds the rest of the query with similar operations.

5.2. Recursion, Quantification, Subqueries, Prolog

Four advanced features can now be presented, but not in detail.

Certain **recursive** queries[9]can be expressed through use of the "Duplicate" menu item, which adds another icon for the same Entity into the workspace and notes that there are multiple occurrences to be handled. Fig. 7 shows how to formulate the query "Which projects follow up the Outils project ?".

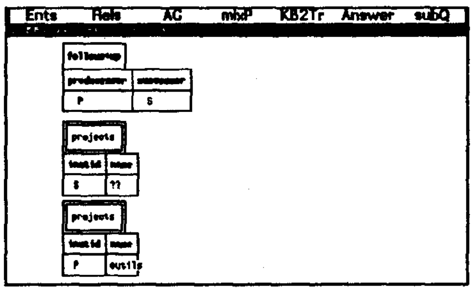


Figure 7: An example of a recursive query

In the schema used here, the "follows_up" Relationship is defined by means of a recursive rule, and the same Entity, "projects", is linked to it twice: once as the "predecessor" role and once as the "successor" role. This entity thus appears in the workspace twice, and the identical variable names ("P" in one case and "S" in the other) indicate which occurrence fills which role.

Logical Variables and Quantification: variable names can be typed in as property values, thus retaining an essential feature of the KB2 linear language. This pos-

[9]See [Cruz 87] for a graphical query language specializing in recursion.

sibility is also used in Fig. 7 where P and S are such variables. Quantification is always implicitly existential if not indicated otherwise. Explicit quantification is dealt with by means of the menu items, "universalQ" and "existentialQ". Choosing either displays the appropriate symbol at the left side of the E-R item icon. **Scope** is expressed by relative spatial position: the left-to-right ordering of quantifiers in the corresponding textual formulae is expressed by the top-to-bottom arrangement in the workspace of the icons representing the entities and relationships. Icons can be moved to wherever desired in the workspace simply by dragging them. Fig. 8 shows two queries (in two workspace fragments). On the left: "Is there a Sun that is used by all employees ?" (i.e. the same Sun is used by everyone); on the right: "Do all employees use a Sun ?" (i.e. each employee may or may not have his/her own individual Sun). The only difference between the two queries lies in the relatives scopes of the quantifiers they contain.

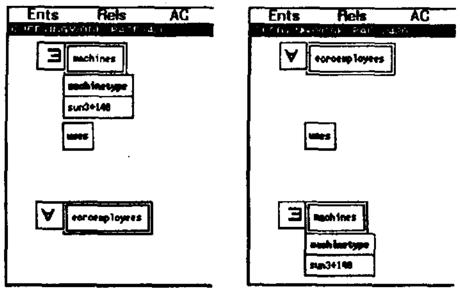


Figure 8: An example with two quantified queries

Subqueries[10]can be derived from workspaces by shrinking them to icons. With the "SubQ" command from the attached menu, icons so created can be copied into a query workspace and used as operands in both atomic and composite conditions, however correctness is checked only at evaluation and not on insertion. Fig. 9 shows an example. It supposes that there already existed an iconised query workspace called "KB_surnames" whose value is the set of the surnames of the members of the KB Group. The example expression asks "What are the ages of the researchers in the KB Group ?".

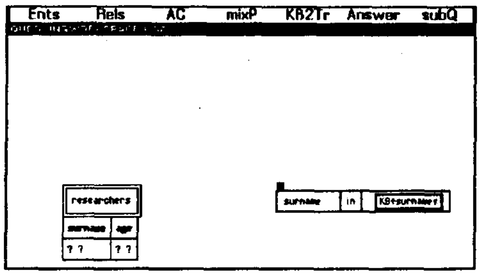
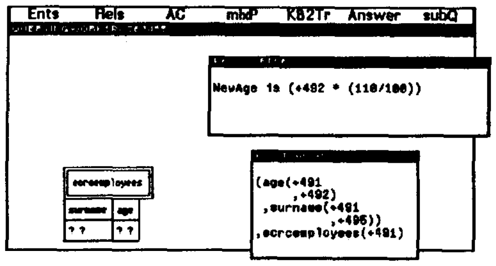


Figure 9: An example of a query containing a subquery

Mixing in Prolog -- creating an expression that contains both KB2 statements and Prolog predicates -- is done by using the "MixP" option in the attached menu. This pops up an editing window into which Prolog can be typed. The code thus entered is either prepended or appended to the KB2 translation of the query expression. Choice of appropriate variable names (thus enabling sharing between the two parts of the final query expression) is the user's responsibility. Variables instantiated during evaluation are treated like property names, and values for them are displayed along with other results. Fig. 10 includes three windows: a workspace, a KB2 translation window (corresponding to only the graphical part of the query), and an editing window for typing in Prolog. The complete query can be paraphrased as "How old would ecrc employees be if they aged by 10 % ?". A mix-in capability of this kind gives Pasta-3's query language a significant degree of on-the-fly extensibility which expert users can exploit to great advantage.



```
NewAge is (+492 * (110/100))
```

```
(age(+491
     ,+492)
 ,surname(+491
         ,+496))
,ecrcemployees(+491)
```

Figure 10: How Prolog code can be mixed in

6. Implementation and Conclusion

We built a throw-away prototype of a subset of the Pasta-3 query language in spring 1988, with the graphical subsystem running on Xerox 1108 hardware and the logical subsystem and the KB2 KBMS installed on a Sun 3-140 (the two subsystems being linked by a TCP connection over the local Ethernet). This version was extensively used for demos and gave rise to many critical observations and comments from people who had been users of KB2's textual language and from software ergonomists from ECRC shareholder companies.

A full-scale implementation begun in summer 1988 is nearing completion. On the graphical subsystem side, all functionality mentioned in this paper is up and running, except Prolog mix-in. The main implementation tool used is the TUBE User Interface Development Environment [Herrm 89, Hill 89]. On the logical subsystem side, coded in Prolog, the core functionalities are finished, however quantification and Prolog mix-in are not yet supported, and the implemented path completion algorithm is not as sophisticated as possible and not yet integrated with the other functionality.

We conclude by summarizing the most important aspects of the work described in this paper. The Pasta-3 query language is based on the **graphical Direct Manipulation** (GrDM) interaction paradigm, which relies

[10]in the same sense as for SQL

on a bit-mapped, multi-window screen and a mouse to implement clickable, draggable icons as the main representation of the query expression. Its purpose is not to invent an nth formal language, but rather to provide a new, sophisticated GrDM editing capability for the already existing formal constructs of the KB2 language. **Cooperative querying** is supported through: handy values, automatic path completion, and the edit-and-reevaluate loop. **Expressive power** is obtained by allowing: arbitrarily complex selections, recursive queries, logical variables and quantification, subqueries, and mixing in Prolog code.

These features of the Pasta-3 query language thus contribute to progress on the aspects of graphical querying where it was most needed. However a major open problem (already mentioned in Section 4.1) is the relationship between GrDM interfaces and DBMS back-ends: to what extent can DBMSs continue to be designed and developed without taking into consideration from the very beginning the requirements of GrDM interfaces ?

7. Acknowledgements

We thank our colleagues Johann Christoph Freytag, Herve Gallaire, Rainer Manthey, and Mark Wallace for their valuable comments on an earlier draft of this paper.

8. References

[Bocca 86] Bocca, J.; Decker, H.; Nicolas, J.-M.; Vieille, L.; Wallace, M. Some Steps Towards a DBMS Based KBMS. In Kugler H.-J. (editor), *Information Processing '86 - IFIP Congress Series*. IFIP, 1986.

[Bose 86] Bose, P.; Rajinikanth, M. An Intelligent Assistant to a Database System. *Texas Instruments Engineering Journal* 4(1):163-169, Jan/Feb., 1986.

[Bryce 86] Bryce, D.; Hull, R. SNAP: A Graphics-based Schema Manager. In *Proc. of the IEEE 2nd Int. Conf. in Data Engineering, Los Angeles*, pages 151-164. 1986.

[Campb 87] Campbell, D. M.; Embley, D. W.; Czejdo, B. Graphical Query Formulation for an Entity-Relationship Model. *Data & Knowledge Engineering* (2):89-121, 1987.

[Catarci 88] Catarci, T.; Santucci, G. Query By Diagram: A Graphic Query System. In *Proc. of the 7th Int. Conf. of Entity Relationship Approach, Rome, Italy*. The ER Institute, 1988.

[Chen 87] Chen,Tao; Wu, Jian-Kang. QPF -- A Versatile Query Language Based on ADTS. In *1987 Workshop on Visual Languages*, pages 199-205. Linkoeping University, August, 1987.

[Cruz 87] Cruz,Isabel;Mendelzon,Alberto;Wood,Peter. A Graphical Query Language Supporting Recursion. In *1987 SIGMOD Conference*, pages 323-330. ACM, May, 1987.

[Dart 88] Dart,Philip;Zobel,Justin. Conceptual Schemas Applied to Deductive Databases. *Information Systems* 13(3):273-287, 1988.

[Du 88] Du, H.; Manoochehr, A. GQL: A Graphical Database Language Using Pattern Images. In *Proc. of the Computer Graphics Int. Conf., Geneva, May*. 1988.

[Egenh 88] Egenhofer, M.J.; Frank, A.U. Towards a Spatial Query Language: User Interface Considerations. In *Proceedings of the 14th Conf. on Very Large Data Bases (VLDB'88), Los Angeles, California*, pages 124-133. 1988.

[Elmas 85] Elmasri, R.; Larson, J. A. A User-Friendly Interface for Specifying Hierarchichal Queries on an ER Graph Database . In *Proc. of the 4th Int. Conference on the Entity-Relationship Approach, Chicago*, pages 236-245. IEEE, October 28-30, 1985.

[Gimn 88] Gimnich, R. Implementing Direct Manipulation Query Languages Using an Adequate Data Model. In *Proc. of the 7th Interdisciplinary Workshop 'Informatics and Psychology': Visualization in Human-Computer Interaction, Schaerding, Austria, May 24-27*. Austrian Computer Society, 1988.

[Goldm 85] Goldman, K. J.; Goldman, S. A.; Kanellakis, P. C.; Zdonik, S. B. ISIS: Interface for a Semantic Information System. In *Proceedings of ACM-SIGMOD 1985 Int. Conf. of Management of Data, Austin, Texas*, pages 328-342. May, 1985.

[Herot 80] Herot, C. F. Spatial Management of Data. *ACM Transactions on Database Systems* 5(4):493-514, December, 1980.

[Herrm 89] Herrmann, M.; Hill, R. D. Abstraction and Declarativeness in User Interface Development -- The Methodological Basis of the Composite Object Architecture. In *IFIP Congress '89*. IFIP, 1989.

[Hill 89] Hill, R. D.; Herrmann, M. The Structure of TUBE -- A Tool for Implementing Advanced User Interfaces. In *EUROGRAPHICS '89*. 1989.

[Hutch 86] Hutchins, E.; Hollan, J.; Norman, D. Direct Manipulation Interfaces. *User Centered System Design*. Lawrence Erlbaum Associates, 1986.

[Kaplan 82] Kaplan, S.J. Cooperative Responses from a Portable Natural Language Query System. *Artificial Intelligence* 19:165-187, 1982.

[Kim 88] Kim, H.-J.; Korth, H. F.; Silberschatz, A. PICASSO: A Graphical Query Language. *Software-Practice and Experience* 18(3):169-203, March, 1988.

[King 84] King, R.; Melville, S. SKI: A Semantics-Knowledgeable Interface. In *Proc. of the Tenth Internat. Conf. on Very Large Databases (VLDB), Singapore*, pages 30-33. August, 1984.

[Kuntz 89a] Kuntz, Michel; Melchert, Rainer. Pasta-3: a Complete, Integrated Graphical Direct Manipulation Interface for Knowledge Base Management Systems. In *to appear in: Proc. 11th World Computer Congress*, pages . IFIP, August, 1989.

[Kuntz 89b] Kuntz, Michel; Melchert, Rainer. Pasta-3's Requirements, Design, and Implementation: A Case Study in Building a Large, Complex Direct Manipulation Interface. In *to appear in: Proc. Engineering for Human-Computer Interaction*, pages . IFIP, August, 1989.

[Kuntz 89c] Kuntz, Michel; Melchert, Rainer. Ergonomic Schema Design, Browsing with More Semantics, and a Graphical Direct Manipulation Query Language: the Pasta-3 End User Interface for E-R DBMSs. In *submitted to: 8th E-R Conf*, pages . E-R Institute, October, 1989.

[Lewis 83] Lewis, J. W. An Effective Graphics User Interface for Rules and Inference Mechanisms. In *Proc. of the CHI'83*, pages 139-143. December, 1983.

[Maier 83] Maier, D.; Ullman, J.D. Maximal Objects and the Semantics of Universal Relation Databases. *ACM Transactions on Database Systems* 8(1):1-14, March, 1983.

[McDon 75]McDonald, N.; Stonebraker, M. CUPID - The Friendly Query Language. In *Proc. of the 1975 ACM PACIFIC, San Francisco*, pages 127-131. ACM, April, 1975.

[Melch 87] Melchert, Rainer. Graphische Unterstuetzung beim Umgang mit Wissensbanken. In *Proceedings of Datenbanksysteme in Buero, Technik und Wissenschaft*, pages . BTN 87, April, 1987.

[Motro 88] Motro, Amihai; D'Atri, Alessandro; Tarantino, Laura. The Design of KIVIEW: An Object-Oriented Browser. In *Proceedings of the 2nd Int. Conf. on Expert Database Systems*, pages 17-31. George Mason University, April, 1988.

[Myers 86] Myers, Brad. Visual Programming, Programming by Example, and Program Visualization: A Taxonomy. In *CHI 86 Conference Proceedings*, pages 59-66. ACM SIGGRAPH, April, 1986.

[Pahwa 85] Pahwa, A.; Arora, A. K. Automatic Database Navigation: Towards a High Level User Interface. In *Proc. of the 4th Internat. Conf. on Entity-Relationship Approach, Chicago, Ill.*, pages 36-43. IEEE, 28-30 October, 1985.

[Poltr 86] Poltrock, S. E.; Steiner, D. D.; Tarlton, P. N. Graphic Interfaces for Knowledge-Based System Development. In *Proceedings of the CHI'86*, pages 9-14. ACM SIGCHI, April, 1986.

[Raeder 85] Raeder,Georg. A Survey of Current Graphical Programming Techniques. *IEEE Computer* 18(8):11-25, August, 1985.

[Reiser 88] Reiser,Brian et al. Visual Programming, Programming by Example, and Program Visualization: A Taxonomy. In *CHI 88 Conference Proceedings*, pages 39-44. ACM SIGGRAPH, May, 1988.

[Shnei 83] Shneiderman, B. Direct Manipulation: A Step Beyond Programming Languages. *IEEE Computers* (16):57-69, 1983.

[Teskey 84] Teskey, F. N.; Dixon, N.; Holden, S. C. Graphical Interfaces for Binary Relationship Data Bases. *Information Systems* 3(2):67-77, April, 1984.

[Udaga 82] Udagawa, Y.; Oshuga, S. Novel Technique to Interact with Relational Databases by Using a Graphics Display. *Journal of Information Processing* 5(4):256-264, 1982.

[Urspr 83] Ursprung, P.; Zehnder, C.A. An Interactive Query Language to Define and Use Hierarchies. *Entity-Relationship Approach to Software Engineering*. Elsevier Science Publishers B.V. (North Holland), 1983.

[Wahls 85] Wahlster, W. Cooperative Access Systems. In *Proc. of the Int. Conf. on Fifth Generation Computer Systems 1985*, pages Vol.1/ pp.103-111. 1985.

[Walla 86] Wallace, M. *KB2: A Knowledge Base System Embedded in Prolog*. Technical Report KB-12, ECRC GmbH, Munich, W. Germany, 1986.

[Willi 84] Williams, M. D. What makes RABBIT run ? *Int. Journal of Man-Machine Studies* 21:333-352, 1984.

[Wong 82] Wong, H. K. T.; Kuo, I. GUIDE: Graphic User Interface for Database Exploration. In *Proc. of the 8th Very Large Databases Conf., Mexico* , pages 22-32. 1982.

[Wu 86] Wu, C. T. A New Graphics User Interface for Accessing a Database. *Advanced Computer Graphics*. Springer-Verlag, Berlin, Heidelberg, New York, 1986.

[Zhang 83] Zhang, Z.-Q.; Mendelzon, A.O. A Graphical Query Language for Entity-Relationship Databases. *Entity-Relationship Approach to Software Engineering*. Elsevier Science Publishers B.V. (North Holland), 1983.

[Zloof 77] Zloof, M. M. Query-by-Example: a data base language. *IBM Systems Journal* (4):324-343, 1977.