The Management of the Ambiguities in a Graphical Query Language for Geographical Information Systems

D. CALCINELLI, M. MAINGUENAUD

FRANCE TELECOM - Institut National des Télécommunications
9 Rue Charles FOURIER F91011 EVRY - FRANCE
+ 33 1 60 76 47 82 + 33 1 60 76 47 80 (fax)
Email : (CALCI, MAINGUENAUD)@FRINT51.bitnet

Abstract

This paper describes the management of the ambiguities which appear while querying a Geographical Database with a graphical query language. The goal is to hide a syntactic query language from end-users who do not usually think in terms of algebraic query language.
Queries addressed to a Geographical Information System are very often divided into two classes : the network management oriented queries and the thematic oriented queries. We briefly present a graphical query language for each class of query. We develop the merger of these two languages in order to attempt to specify a unique graphical query language for geographical databases. We point out the various ambiguities due to the merging phase. These ambiguities mainly derive from the semantics of the operators and from the different levels of manipulation (logical or geometrical). We present the solutions we adopted in the context of the CIGALES* project.

Keywords : Geographical Information System, Graphical Interface, Data Base, Data Manipulation Language

1. Introduction :

Various domains such as urban planning, remote sensing and network management use geographical data. The applications principally use a Geographical Information System (GIS) [16].
The issue we focus on in this paper is the user-interface of such a system in a Data Base Management System context [17]. The end-users need not to be familiar with data base

* CIGALES stands for Cartographical Interface Generating an Adapted Language for Extensible Systems

management concepts to formulate a query. Workstations offer the opportunity to define new query languages based on graphical (or visual) manipulations [9, 15]. We call a language "graphical" whenever the semantics of the operators are expressed with a drawing (and not just by a click-in-a-box philosophy in order to build an alphanumerical query). The main advantages of this approach are : (1) the user-friendliness of the graphical manipulations, (2) the expression of the graphical query with a Query-By-Example philosophy and (3) the use of a mental model of the manipulated objects (allowing logical DBMS data model and physical data structure storage independences).

Traditionally, geographical applications distinguish between network-oriented queries and thematic-oriented queries. This distinction was due to the lack of common data structures, data manipulation operators and an unified Data Manipulation Language (DML). The main consequence has been the development of several different DMLs, i.e. [8], [7]. But these widely discussed languages follow an algebraic approach (except [5]). In order to avoid the dichotomy of the query languages, the scope of this paper is to present the unification of two graphical DMLs. Grog [10] is a graphical network-oriented DML and Cigales [12] is a thematic-oriented DML. The expression of a query with a drawing may produce misunderstandings between the end-users and the GIS query evaluator. These ambiguities which can occur while defining a query, are due to the 2D nature of the query representation. The merger of two graphical languages implies the definition of a policy to resolve the ambiguities .

The paper is organized as follows : section 2 briefly presents the two graphical DMLs, Grog and Cigales; section 3 presents the ambiguities born from the merger of these two languages; section 4 presents the solutions we adopted to solve these ambiguities; section 5 presents the implementation details; and section 6 deals with the conclusions and further work.

2. Grog and Cigales :

This section is dedicated to the presentation of two graphical languages. A more complete presentation can be found in [10] and [12].

2.1. Grog : a DML for network-oriented queries :

Four main classes of typical queries addressed to a network-oriented information management system can be observed : (1) path evaluation i.e.going from one place to another under constraints, (2) intersection of paths i.e. common sub-path(s) between two

paths, (3) inclusion of path i.e. path evaluation with a specific sub-paths and (4) node manipulations i.e. common places between two paths.
Some of these typical recursive queries can be modelled with Horn clauses [3]. But Horn clauses are not well suited as a user-interface. Horn clauses are not a very user-friendly language for anyone who is not used to recursive concepts. Furthermore, they cannot model some classical geographical queries such as : "What are the common parts of the paths via the TGV railways between Paris-Lyon and Lille-Geneva ?" (Class 2), or "What are the paths between Paris and Nice via the motorways between Lyon and Marseille, without any cycle ?" (Class 3).

Data : Data are modelled by directed graphs. A directed graph G is represented by G(N, E) where N is a set of nodes and E is a set of edges between two elements of N. Nodes and edges are labelled.

Manipulations : The manipulations are based on the evaluation of a path. This operation is a well-known data base problem (transitive closure of a relation) i.e. [1, 2]. The manipulations are defined over the edges, over the nodes and the edges and over the nodes.
The manipulation of the edges are : the Direct Link -Π_D- between two nodes (i.e. going from one place to another in one step) which corresponds to evaluate the successors of a node in a graph, the Transitive Link -Π_T- between two nodes (i.e. classical graph traversal symbolized by "+") which corresponds to evaluate a path between two nodes in a graph, the Intersection of paths (edges) -Π_{is_e}- between two paths which corresponds to evaluate the common edges of two paths, and the Inclusion of a path (edges) -Π_{ic_e}- into another path which corresponds to define a path with a specific sub-path.
The manipulation of nodes and edges is the Inclusion of nodes -Π_{ic_ne}- in a path (i.e. going from one place to another with some specific stop places).
The manipulations of nodes are : the Intersection of nodes -Π_{is_n}- (i.e. common places between two paths), and the Inclusion of nodes -Π_{ic_n}- (i.e. paths with the same stop places whatever the ways of joining them are).

Queries : A graphical query Q defined on a graph G is a set of labelled and oriented graph-like structures. The labels of the nodes can be variables or constants. Three types of edges are available to model the basic operators (link, inclusion, intersection) :

Link 

Inclusion 

Intersection 

These edges follow the same rules : (1) an edge is oriented, (2) an edge is a binary operator (one initial point, one end point), (3) an edge represents the results of a sub-query (i.e. the evaluation of the operator between the initial point and the end point), and (4) an edge represents a set of acyclic graphs (or paths).

Example : The query : "I would like to go from Paris to Nice with a stop in Lyon via the TGV railways" is expressed by the figure 1 :

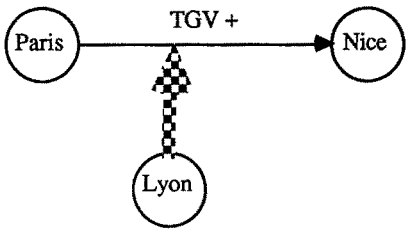


figure 1

The nodes labelled Paris, Lyon and Nice represent the three towns. The thin arrow between Paris and Nice represents the path between these two towns. The label + asks for the evaluation of a path (Π_T). This path may be seen as a graph G (N, E). The arrow between Lyon and the thin arrow asks for the node labelled Lyon to be included in the set of nodes N of G.

2.2 Cigales : a DML for thematic-oriented queries :

In this language, two basic objects are defined : the line and the area. The user draws a pattern according to his mental model of the real world. In his mental model, he will for example see a road as a line (even if the physical representation in the database is an area : i.e. large scale), and a town as an area.

Data : The use of a mental model allows to be independent from the logical DBMS data model. We choose to represent the data with a simple hierarchy (as the simple inheritance object oriented data models do). Each "object" is characterized by a set of attributes. The query evaluator is in charge of translating the selection criteria defined on these "objects" into logical DBMS data model dependent queries.

Manipulations : A set of operators has been defined in the Cigales language to build a query. The classical geometrical operators are available : inclusion, intersection, adjacency. Special operators like path and the euclidian distance between two objects are also available.

Query : To build a query, the user selects basic objects, i.e. lines and areas, associates the semantics to these basic objects (i.e. a line corresponds to the national road), and assembles

these basic objects by applying an operator. A graphical editor (figure 2) is dedicated to the construction of a graphical query :

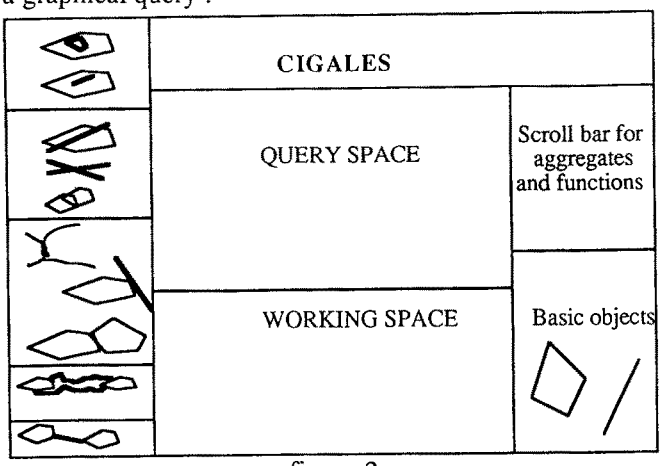


figure 2

Icons model the available operators (left hand side of the graphical editor). Basic objects appear in the right hand side of the graphical editor. The data model is used to notify the alphanumerical part of a graphical query and is automatically invoked by selecting a basic object.

Two spaces are defined : the working space and the query space. The first one is used to build a sub-part of the query : in this space, the user defines two objects and applies an operator. He then validates this part of the query which appears in the query space. Objects defined in the working space may be new objects or already existing objects selected by the user in the query space.

Example : The query : "I would like to know the national roads crossing the TGV railways joining Paris to Lyon". To build such a query, the user first selects the basic object area, and associates the semantics : town of Paris. This object then appears in the working space. The same manipulation is performed about the town of Lyon. The path operator is then applied to these objects with the semantics "TGV railways", to compute all the paths using the TGV railways from Paris to Lyon (several TGV railways may contribute to this path). Then, the user validates this part of the query which is moved into the query space. Now, he wants to define the intersection between the T.G.V. railway paths and some national roads. The basic object line is selected with the semantics national road. The TGV railway path from Paris to Lyon is selected in the query space and then appears in the working space -but the opposite order (1) the TGV railways (2) the national road may have been performed since the intersection is a commutative operator. The intersection operator is applied between these objects. This part is also validated and appears in the query space. The whole query is validated and can be resolved. Figure 3 is the graphical representation of this query available in the query space :

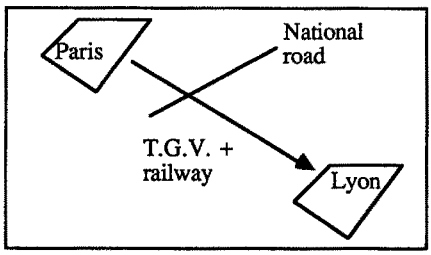


figure 3

3. Ambiguities :

Unfortunately each of these two languages cannot be considered as a complete language for GIS. Grog is not well suited to model "area" manipulations and Cigales does not manage very well "path" manipulations.
A natural evolution is to merge Grog and Cigales. Cigales is the base for the new improved language. Using a graphical query language or while defining this merger, various classes of ambiguities appear. A graphical query is said to be ambiguous whenever several interpretations could be performed. This section is dedicated to the presentation of three classes : the visual semantics, the level of abstraction and the query labels.

3.1 The visual semantics :

This ambiguity comes from the graphical query language orientation. The central notion of a query is the composition of the operations. The semantics of each operator is clearly defined and well known (i.e. intersection, inclusion) but the composition of these operators may lead to misunderstanding. See as an example figure 4 :

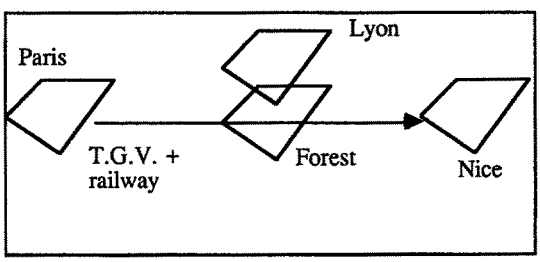


figure 4

A naïve interpretation is : A path, between Paris and Nice via the TGV railways, crosses the forest around the town of Lyon. The problem here is : Does the query forbid the TGV railways to cross the forest part of Lyon (the common part of the "area" forest and the "area" town of Lyon) ?

3.2 The level of abstraction :

Network-oriented query languages and thematic-oriented query languages do not manage the same level of semantics. For example, Grog is oriented toward a logical representation. An edge of the data model between two nodes (i.e. towns) does not refer to any physical representation. This edge is an abstraction of the physical link (as Abstract Data Types do). This logical representation hides the physical coordinates which refer to the real link.

On the contrary, for example, Cigales is much more oriented toward a symbolic graphical representation. An area is considered as "unique" even if this "area" may be physically in fact constituted by a set of "areas" (i.e. a country with some oversea lands). All its operators refer to graphical manipulations (even if the user does not have any notion of coordinates).

Two kinds of ambiguity may appear here : the difference of semantic level and the logical link between the operators.

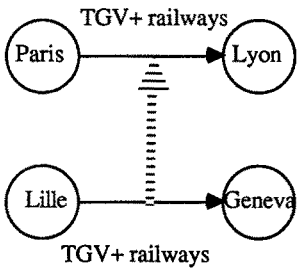


figure 5

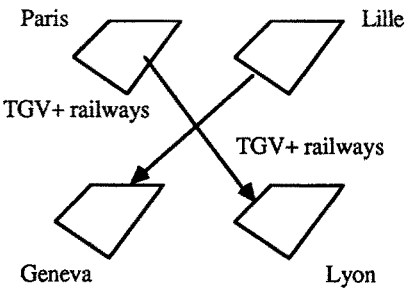


figure 6

Figure 5 and figure 6 are two representations with Grog and Cigales which may signify an identical query : "What are the common part(s) of the paths via the TGV railways between Paris-Lyon and Lille-Geneva ?".

Unfortunately the results of such queries with the formalism of figure 5 and figure 6 are different. The result of the query in figure 5 is the TGV railway paths with their common part(s) and their exchange points (a connection is possible between the two TGV trains - this is due to the logical representation of the network: a graph). But the result of the query in figure 6 is a super-set of the result of figure 5. In this query all the intersections between the two paths are obtained (even if a connection is not possible i.e. a TGV line crosses another over a bridge - this is due to the 2D geometrical representation of the networks- nowadays it is very difficult to jump from one train into another over a bridge !).

The Query-By-Example philosophy breaks the logical link between the operators. This separation cannot be avoided at the query resolution model level. The logical link is implicit in the network-oriented query language operators. Let us see figure 1 as an

example of the query : "I would like to go from Paris to Nice via the TGV railways with a stop in Lyon".
This query cannot be expressed with the same semantics in Cigales because figure 7 which may seem equivalent :

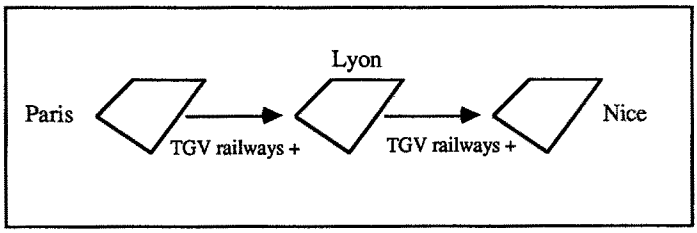


figure 7

does not define a link between the two path operators (path Paris-Lyon and path Lyon-Nice - see figure 8 as an example of data modelled by a graph). This freedom may introduce the apparition of common places (nodes or links) between the two paths which is not allowed in Grog (see rule 4 for the definition of a Grog query : Paris-Nevers-Lyon-Nevers-Nice).

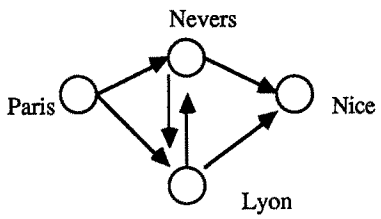


figure 8

3.3 The query label :

Two kinds of ambiguity appear while defining the label of the "objects" : the notion of order and the notion of identity of object.

The ambiguity of the order is illustrated with the following example : "I would like to go from Paris to Nice via the TGV railways with a stop in Geneva and in Lyon" if we compare it to : " I would like to go from Paris to Nice via the TGV railways with firstly a stop in Geneva and secondly a stop in Lyon".
A single "object" cannot represent the two objects Lyon and Geneva. The labels such as :

$$(1)\ \text{Town} = \text{Lyon} \vee \text{Town} = \text{Geneva}$$

$$\text{and}\ \ (2)\ \text{Town} = \text{Lyon} \wedge \text{Town} = \text{Geneva}$$

do not fulfill the conditions since with the label (1) a path with a unique stop (i.e. in Lyon) will verify the query and with the label (2) a town cannot verify two selection criteria (the name of a town is supposed to be unique). The Query-By-Example philosophy may lead

the user to specify the first stop and then the second stop, but what happen when the order is not important or when the user does not know the order of the visited towns ?

The ambiguity of the identity of object can be illustrated with the following example [13] : "I would like to know the roads adjacent to a lake but the road has to be the national road 13 whatever the function of the lake is or the lake must have a function of leisure base whatever the type of the road is.
Let figure 9 represents the basic drawing without any alphanumerical label :

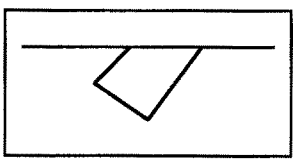


figure 9

To keep user-friendliness of the graphical language the query has to be expressed with only one drawing. The alphanumerical selection criteria has to be defined within the objects. Defining two graphical queries (see figure 10) would not follow the graphical philosophy.

figure 10

The query cannot be modelled by :

Adjacent (road [Type = Road $\wedge$ Name = NR13],
lake [Type = Lake $\wedge$ Function = Leisure base])

where Adjacent is the name of the operator and [...] represents the alphanumerical selection criteria since only national road 13 will be taken into account.

Adjacent [({Type = Road},{Type = Lake $\wedge$ Function = Leisure base}) $\vee$
({Type = Road $\wedge$ Name = NR13}, {Type= Lake})]
(road, lake)

corresponds to the query. One can notice that the label is defined at the operator level (and not at the "object" level) and the position of the selection criteria corresponds to the position of the operandi.

4. Proposed solutions :

This section is dedicated to the presentation of the solutions we adopted to solve these ambiguities in the context of the CIGALES project. Two principles may be studied : (1) as soon as an ambiguity is detected, a feed back with the user is proposed in a natural-like language or an internal formalism, (2) default semantics are defined for each case of ambiguity. The first solution introduces various interactions with the user and may be boring even if the interaction is performed in a natural-like language. The second solution (the principle we adopted) increases the complexity of the resolution model, but simplifies the user's actions.
Choosing a solution among several propositions must not forbid the expression of the rejected propositions. The merger between Grog and Cigales introduces some changes in the Cigales interface. Each subsection presents the rule (default semantics) we chose and the consequences in the user interface for the three classes of ambiguities we develop in this paper : the visual semantics, the level of abstraction and the query label.

4.1 The visual semantics :

The independence between operators is defined to avoid conflicts during the composition of the operators. An operator is defined with two operandi. The construction of an operator takes as first and second operandum a basic object, an already defined object or the result of the application of an operator. This recursive definition has no limit (except the fact that a user cannot interactively define an infinite composition of operators).
The independence means there exists no link between the operators unless this link has been explicitly specified. This link is established by selecting an object in the query space instead of taking one of the basic objects (in the right hand side of the graphical editor - see figure 2) or by selecting an object using the selective union mode.

Consequences upon the interface : the selective union mode

By default the independence mode is on. This absence of link between the operators means for example in figure 4 that the intersection operator between the path and the forest does not interact with the intersection operator defined between the town of Lyon and the forest.
A query is modelled by a functional expression [12]. Figure 11 is part of the internal representation of a graphical query (the graph structure is a partial representation of the optimized functional expression). The leaves represent the basic "objects" involved in the query and the non-leaves nodes are the operators :

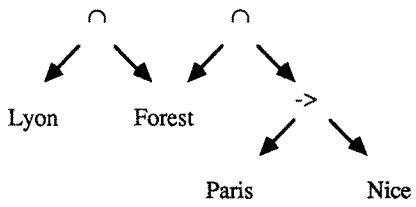


figure 11

where ∩ represents the intersection operator and -> represents the path operator. One can see that no interaction is defined between the two intersection operators.

In the selective union mode, all the atomic objects are built using implicit Δ and ∪ operators. The Δ operator (geometrical difference) is not available at the user interface level but is defined as an implicit operator to build atomic parts of an object. The union operator is available at the user interface level but is invisible since it is performed as soon as several clicks have been performed on already existing objects in the query space.

The selective union mode allows the decomposition of an object into several atomic parts. Following our example in figure 4, the basic object forest is decomposed into two sub-parts : part 1, the forest which is part of the town of Lyon and part 2, the complementary portion of the forest (defined with a Δ operator). The implicit union operator allows building of more complex objects in case of several sub-parts. A click (in the selective union mode to define the involved objects of the intersection operator) in the part of the forest which is not related to the town of Lyon constructs a new object built with the geometrical difference .
The query can now be partially modelled by (figure 12):

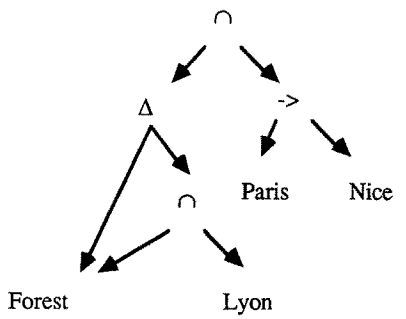


figure 12

One can notice that an interaction is now defined between the two intersection operators.

4.2 The level of abstraction :

The common basic operations between network oriented queries and thematic network oriented queries are the intersection and the inclusion. A user defines a query

following his mental model (using the basic objects of Cigales where the node concept of Grog is represented by an area and the edge concept is represented by a line or the path operator). So, to fix the semantics of the operators between two objects, an assumption guides our choice. This sub-section details for the intersection and the inclusion operators, the assumption (when necessary) and the semantics of the default operator following the types of the involved objects (line and area).

Intersection* :

Between two lines : The operator exists in Grog as well in Cigales. The assumption we retain is the following : "Whenever an intersection appears between two lines of the mental model, it should be a network oriented query". So the Grog operator Π_{is_e} is retained for this operation as the default operator.

Between an area and a line : No confusion may arise here since in Grog there does not exist an intersection operator between an area and a line. So the Cigales operator $\cap$ is retained for this operation as the basic operator.

Between two areas : A confusion may arise at this level but, in fact, the Grog operator (Π_{is_n}) does not provide in its semantics more information than the Cigales operator ($\cap$). A difference is introduced at the resolution model due to the complexity of the operation [14]. The Grog operation can be performed using a relational DBMS [11] but the Cigales operator needs a much more complex operator (in time). So for optimization purposes a difference, invisible from the user's point of view, is introduced whenever two Grog objects are involved (replacing $\cap$ by Π_{is_n}).

Inclusion :

Between a line and an area : No confusion may arise since this operator does not exist in Grog. So the Cigales operator, inclusion, is retained for this operation as the basic operator.

Between two areas : A confusion may arise at this level but, in fact, here the Grog operator (Π_{ic_n}) does not provide in its semantics more information than the Cigales operator. A difference is introduced at the resolution model due to the complexity of the operation. The Grog operation can be performed using a relational DBMS, but the Cigales operator needs a much more complex operator (in time). So for optimization purposes a difference, invisible from the user's point of view, is introduced whenever two Grog objects are involved.

---

* The intersection operator is commutative so only half of the combinations are studied.

Between an area and a line : No confusion may arise here since this operator does not exist in Cigales. So the Grog operator Π_{ic_ne} is retained for this operation as the basic operator.

Between two lines : As in the previous case Cigales does not provide such an operator. So the Grog operator Π_{ic_e} is retained for this operation as the basic operator.

Figure 13 presents the interpretation of the operators depending on the user's mental model of the "objects".

		Line	Area
Intersection	Line	Π_{is_e} / $\cap$	$\cap$
	Area	$\cap$	$\cap$
Inclusion	Line	Π_{ic_e}	$\subseteq$
	Area	Π_{ic_ne}	$\subseteq$

figure 13

The consequences upon the interface is the creation of a new icon : The ambiguity may appear during the definition of the intersection between two lines. To change the default semantics, the choice of the geometrical mode is performed by a user's voluntary action (by selecting it in the functions scrollbar menu).
To deal with operators which are not defined in Cigales (i.e. inclusion of an area in a line) a new icon is offered to symbolize this operation in the working and query spaces. But the icons of the operators are not changed. Figure 14 presents the new icon such as it appears in the working or query space.

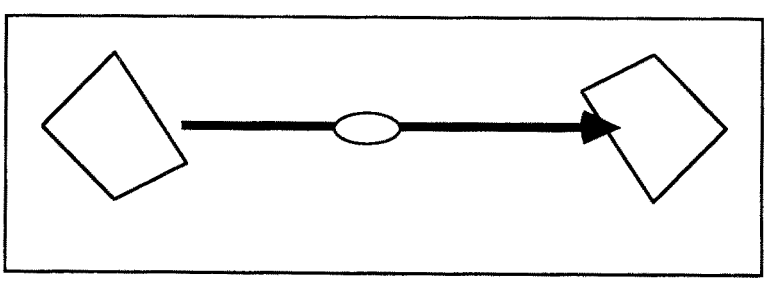


figure 14

Using the selective union mode with the new icon allows dealing with the inclusion of a line in a line operator which does not exist in Cigales. This operator can be defined with an area and the sub-part of the path defined in the new icon.

4.3 The query label :

The introduction of the selective union mode and of the new icon allows managing the ambiguity due to the order. If the order is known, the recursive application of the selective union mode and the decomposition of the new icon give an opportunity to define the correct sub-path. If the order is not known, the implicit union mode allows selecting the entire path and requires that "places" must belong to this path, whatever the order of the visited "places" is. The basic semantics are equivalent in both cases, so the icon is the same (see figure 14).

The notion of identity requires the postponement of the definition of the labels. To do so, an option of the data model selection criteria definition gives an opportunity to refuse the definition while selecting a basic object. The information are entered during the definition of the operator label.

The consequences upon the interface is the postponement of the alphanumerical labels : The management of the order does not introduce any modification. But the definition of the operator label may need to define various selection criteria for two objects (involved with this operator). Therefore the semantics of the query are fully-user-dependent. The end-user is in charge of the coherence of the labels in such a query.

5. Implementation :

The user interface is currently under implementation on a Sun workstation, with the application framework Aïda [6]. Aïda allows buttons, icons and vertical scrollbars to be used as images, activated with the mouse. The associated actions are developed with the Le-Lisp language [4].

In this section we describe the user interface, its environment, an example of query construction; we conclude with the actual restrictions of the prototype.

5.1 The user interface description :

The user interface consists of four zones (see hard-copies number 1 to 4).

The middle of the screen is dedicated to the query definition part. The part is divided into two spaces : the query space (on the top) and the working space (on the bottom). They are

two Le-Lisp virtual windows, providing graphic and bitmap function management. The first one contains the query at each step of its construction. The second one is used to define objects which are combined by the application of an operator.

The upper zone presents the name of the project (CIGALES) and provides six functions. On the right side, three buttons named Save, Help and End act on the application level. They respectively save a query (saving its internal modelling and its graphic representation), provide on-line help, and quit the application. On the left side, Request and Working are two vertical scrollbars, when Selection is a button. The Working menu provides the following features : Valid (validation of the working space, drawing its sub-query in the query space), Undo (cancelation of the last action in the working space) and Clear (reset of the working space). The Request menu provides the following features : Valid (validation of the global query, allowing its evaluation), Undo (cancelation of the last validated sub-query) and Clear (reset of the query space). The goal of the Selection button is to allow a selection of an object from the query space, in order to use it in the working space instead of selecting a basic object.

The column on the right side of the screen supplies two scrollbar menus and two icons. The scrollbar menus provide Utilitarians (i.e. zoom, specific display, semantic modification for the intersection operator) and Aggregate functions (i.e. min, max). The two icons represent the basic objects for the graphical query language : a line and an area.

The column on the left side of the screen supplies the graphical operators (inclusion, intersection, adjacency, path and euclidian distance) with five icons. For example, the second icon represents the three symbols for the intersection operator. Those symbols are just a help for the user. He does not need to click on a specific symbol, but just click on the generic icon.

5.2 The user interface environment :

We may think about the data model and the internal query modelling as the input and output of the user interface.

Any data model call induces a specific window display. Within it, we principally provide a hierarchical data description, represented by a tree. Any selection of a new basic object (line or area) induces automatically a data model access. Nevertheless, the selection of the path operator also induces a data model access, in order to qualify the path (which is a list of line basic objects). When the data model is invoked, the user may qualify the object, or choose to postpone the definition. The qualification is performed when the operator is selected, inducing therefore a new data model access while defining the operator.

The internal queries expressions may be classified into three classes. The first class contains the orders for the application-management level, such as connection, deconnection or reset, invisible from the user's point of view. The second class contains the orders for the screen-management level, such as zoom or point location, defined upon the results of a query. The third class allows composing the operators during the graphical construction of the query. Each sub-query defined in the working space is associated with a partial functional expression. The functional expression is constructed step by step during the graphical query build-in. This construction mode induces duplicated sub-expressions, which are deleted by the query resolution engine, in charge of optimizing the query at a logical level. Anyway, these duplications are necessary during the construction of the query, in order to allow an easy management of the undo facility at any step of the construction.

5.3 A graphical query build-in :

This section is dedicated to the evolution of the user interface, following step by step the definition of the query : "Which national roads cross the Paris-Lyon TGV Railway ?".

In order to construct the sub-query for "the Paris - Lyon TGV Railway", the user first selects the area basic object, by choosing the corresponding icon on the right column. He next qualifies this area since the data model has been automatically invoked. The internal representation of the label is the selection criteria [Type = town $\wedge$ Name = "Paris"]. The object (with part of its label) is then displayed on the working space, and a functional sub-expression, formed with an object identifier and the label, is associated to the object. By the way, the "Lyon" area is displayed on the working space, and a corresponding functional sub-expression is associated with the object. At last, the user chooses the path icon operator and qualifies this path [Type = railway $\wedge$ Name = "TGV"] with a data model access. So, the basic objects and the icon representing the result of the operation are available in the working space (see hard-copy number 1). At this stage, the user can valid the sub-query with the Valid option of the Working menu. Thus, the sub-query appears on the query space and the working space is cleared, as shown in hard-copy number 2.

The user selects now the line basic object, by choosing the corresponding icon on the right column, and he qualifies it [Type = roads, Name = "National road"]. The creation of the associated functional sub-expression is performed. The user then chooses the Selection button. So, he can select the dash modelling the TGV Railway available in the query space, which appears then on the working space. He needs now to select the graphical intersection operator (the Cigales one). Since the default intersection operator is logical (the Grog one), the user has to explicitly change its semantics. That is why the graphical intersection operator is available within the Utilitarians scrollbar menu. That operator appears in the working space by the drawing of the intersection of lines, as shown in hard-copy number 3. It also induces the construction of the functional sub-expression representing this

intersection. With the validation of this sub-query with the Valid option of the Working menu, the screen becomes as shown in hard-copy number 4, expressing the desired query. The conclusive functional expression includes (1) the functional sub-expression of the path with the identifiers and the criteria of the basic objects (two areas), (2) the functional sub-expression of the graphical intersection operator (an operator and a line). At the end, the global query is validated using the Valid option of the Request menu.

5.4 The actual restrictions :

The main restrictions on the actual interface implementation are the introduction of the union concept and the management of the complex queries.

The selective union operator and the named union operator are not yet available. As a matter of fact, the Selection button would have to represent the named union operator. Indeed, it is just a unique selection button, without any union notion. This evolution is not very difficult to handle since this does not involve graphical manipulations.

Complex queries, with numerous basic objects (imagine an area with few bordering basic objects and as much intersecting basic objects) are not expressible. The evolution is much more complex to handle since it is strictly graphic, depending on the screen and the basic objects dimensions. In any way, the interface allows to define simple queries (most of the queries needed by the end-users).

6. Conclusion :

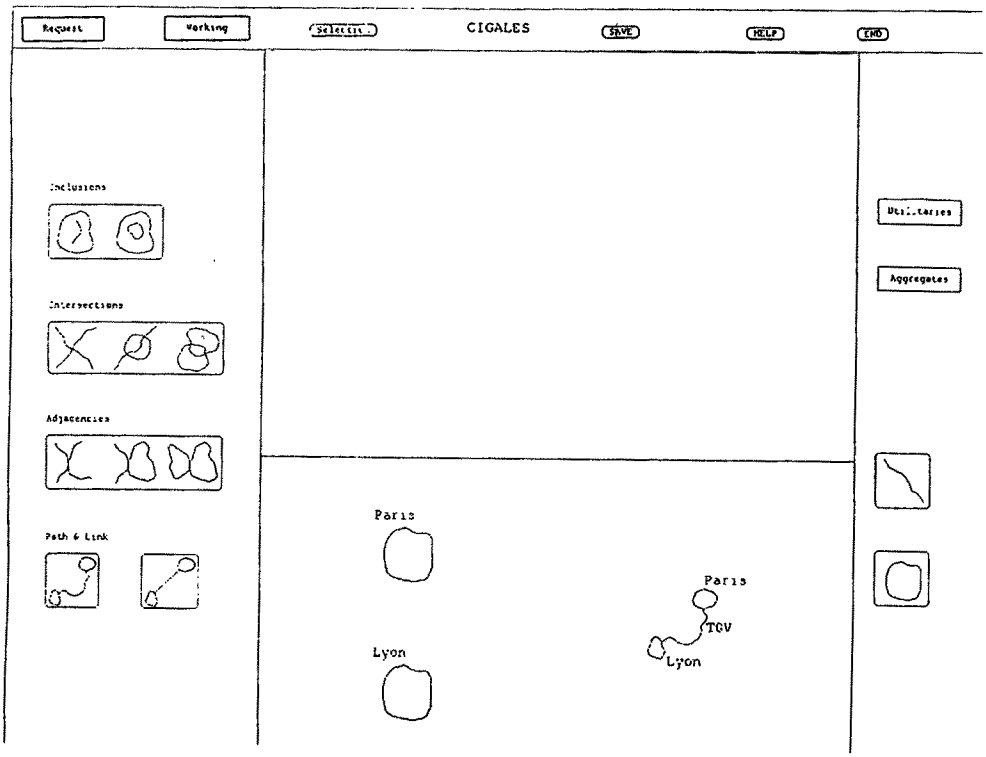
Substantial work has been done in the GIS (Geographical Information Systems) field to improve the data structures, the operators, or the query languages. The user interface is very important in such a field because the users are not willing to make any effort to deal with specific computer problems. Even if the domain is far from being mature i.e. the optimization of the query resolution model and the non-application-dependent query language, GIS can take full advantage of powerful graphical workstations.
The GIS query languages present a dichotomy between network-oriented queries and thematic-oriented queries. The scope of this paper is to introduce a merger between these two philosophies, and we develop the management of the ambiguities which can occur while defining a query using a graphical query language (or a visual query language [15]).
The ambiguities we present in this paper may be divided into three classes : the visual semantics, the level of abstraction and the query label. We explain the solutions we adopted in the context of the CIGALES project in order to attempt to define a complete graphical

query language for GIS. These solutions are based on the independence of the operators, the default semantics of the operators, and the introduction of the operator labels.
Our next work is to introduce the negation concept. This is a very hard problem for many reasons : (1) its introduction may alter the safety of the query in a database context for the query resolution model, (2) its graphical representation is very difficult to manage in the context of a query language following a Query-By-Example philosophy since by definition the "objects" do not exist.

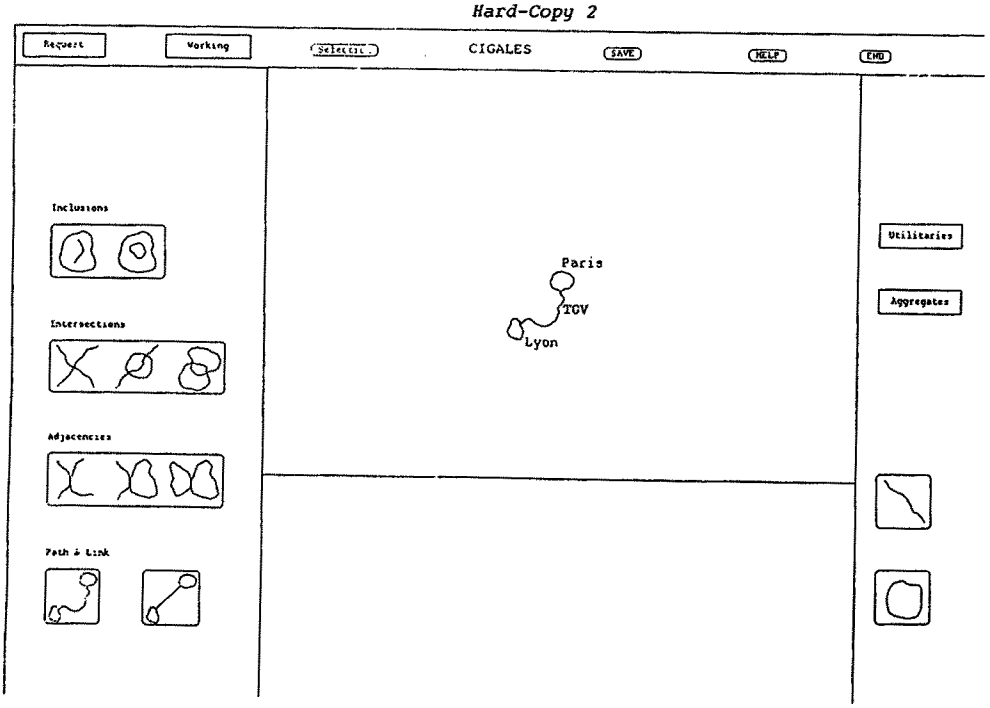
REFERENCES :

1. Aho A., Ullman JD : Universality of Data Retrieval languages, Proceeding of the ACM POPL Conference, San-Antonio, USA, Jan 1979
2. Bancilhon F., Ramakrishnan R. : An amateur's Introduction to Recursive Query Processing Strategies, Proceedings of the ACM SIGMOD Conference, Washington, USA, May 1986
3. Ceri S., Gottlob G., Tança L. : Logic Programming and Databases, Springer Verlag, 1989
4. Chailloux J. : Le-Lisp de l'INRIA, Version 15.22, Jan 1989
5. Cruz I, Mendelzon A.O., Wood P. : Graphical query language supporting recursion, SIGMOD Conference, San-Fransisco, USA, 22-29 May 1987
6. Ilog : Aïda, Reference Manual, 1988
7. Franck A. : MAPQUERY : Database Query Language for retrieval of geometric Data and their graphical representation, Computer Graphics, Vol 16, n° 3, July 1982
8. Jungert E. : Inference rule in a Geographical Information System, IEEE Workshop on Language for Automation, New-Orleans, USA, November 1984
9. Kim HY and al : PICASSO : A Graphical Query Language, Software-Practice and Experience, Vol 18(3), 169-203, March 1988, Ed. J. Wiley and Sons Ltd
10. Mainguenaud M. : GROG : Geographical Queries using Graphs, Advanced Data Base System Symp., Int. Proc. Japan Society, Kyoto - Japan, 7-8 Dec. 1989
11. Mainguenaud M. : Is an Extended Relational Data Base Management System Powerful Enough to Deal with Network Oriented Queries, European Geographical Information System Conference, Amsterdam - The Netherlands, 9-13 April 1990
12. Mainguenaud M., Portier M.A. : Cigales A Graphical Query Language for GIS, 4th Spatial Data Handling, Zurich - Switzerland, 22-28 July 1990
13. Ooi B.C., Sacks-Davis R. : Query Optimization in an Extended DBMS, Foundations Of Data Organization Conference, Paris, France, June 1989
14. Preparata F.P., Shamos M.I.: Computational Geometry : an Introduction, Springer-Verlag, 1985
15. Shu N.C. : Visual Programming, Van Nostrand Reinhold Cie, New York, 1988
16. Smith T.R., Menon S., Star J.L., Ester J.E. : Requirements and principles for implementation and construction of large scale GIS, International Journal of GIS, Vol 1, n° 1, 1987
17. Ullman J.D. : Principles of Database, Computer Science Press, 1980

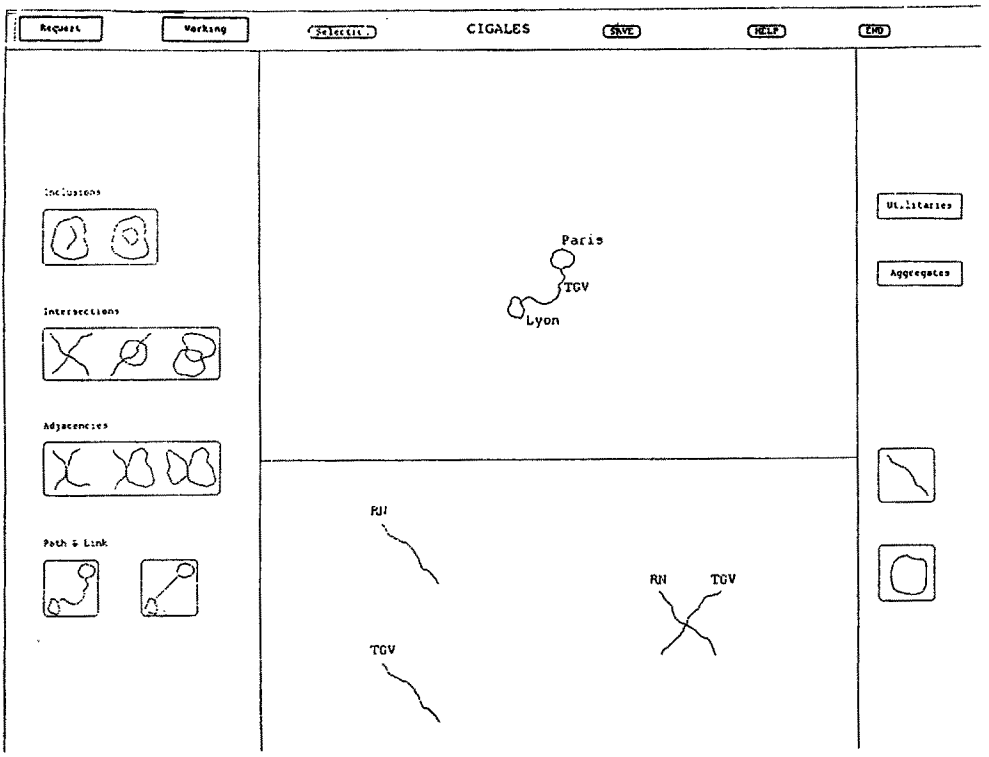
Hard-Copy 1



Hard-Copy 2



Hard-Copy 3



Hard-Copy 4