The Management of Spatial and Temporal Constraints in GIS using Pictorial Interaction on the Web

Fernando Ferri[1], Patrizia Grifoni[1], and Maurizio Rafanelli[2]

[1] IRPPS-CNR, via Nizza 128, 00187 Roma, Italy
{f.ferri, p.grifoni}@irpps.cnr.it
[2] IASI-CNR, viale Manzoni 30, 00185 Roma, Italy
{rafanelli}@iasi.cnr.it

Abstract. This paper proposes and discusses an eXtended Pictorial Query Language (XPQL) for geographic data, and its possible use on the Web. This query language is mapped in Geographic Markup Language (GML 3.0), which is used for transport to the Web and storage of geographic information, including spatial and non-spatial properties of geographic features, as well as the temporal and topological properties of such information. The XPQL algebra is described and an example of this language is given using the implemented prototype. The different phases of query resolution are also explained, as well as the method used to solve ambiguities arising from drawing the different features of the pictorial query. Particular attention is paid to queries that include temporal constraints, and the way to express these in a pictorial environment.

1 Introduction

Many researchers have recently focused their attention on different approaches to express queries regarding geographical data. The evolution of visual query languages has led to the proposal of pictorial query representation in which both spatial and temporal variables are present. In reality, these are "always" present in geographical data (*where* and *when*), but the temporal dimension of the query is often ignored. In addition, computer networks and distributed computing technology have transformed many computer-based applications from the traditional stand-alone mode to the networked mode. This paper, starting from the analysis of query languages for distributed GIS, proposes and discusses an eXtended Pictorial Query Language (XPQL) for geographic data including spatial and temporal relationships, and its possible use on the Web. This language is mapped in Geographic Markup Language (GML 3.0) [1], which is used for exchanging and storing geographic information via the Web.

Various proposals for Visual Query Languages for Geographical Data have recently been made. Some of these consider only limited spatial relations. For example, in [2] the proposed language has the main goal of finding directional relationships in symbolic images. In contrast, SVIQUEL [3] applies techniques of direct manipulation to spatial data exploration and supplies tools to visually query such data. A relevant proposal regards Cigales [4] and its extension LVIS [5], which are the first pictorial query languages allowing the user to draw a query. Specifically,, Cigales is based on the idea of expressing a query by drawing the pattern corresponding to the result desired by the user. It uses a set of active icons which are able to model the needed graphical forms, that is, the geometrical objects polyline and polygon (point is not considered), and the operations carried out on these objects (intersection, inclusion, adjacency, path and distance). The graphical forms and icons conceptualizing the operators are predefined. Other query languages use the blackboard metaphor, as in Sketch [6] and Spatial-Query-By-Sketch [7]. The user is free to draw by hand shapes or symbols on the screen to express an operator or an object.

In visual query languages a query can lead to multiple interpretations and thus ambiguities for the system and user. One of the main reasons is that a unique working space is often used to represent and express different kinds of information. Another is the different approach of the user in formulating his query with respect to the analysis that the system carries out on that query.

The eXtended Pictorial Query Language (XPQL) resolves limitations due to such ambiguities, thus improving and simplifying human-computer interaction.

Literature provides us with some significant examples of query languages defined for XML [9] [10]. Among the more important papers, XQL [11] enables the user to query XML documents one at a time, but it does not solve the problem of querying more than one such document instance at a time. In contrast, XML-GL [12] facilitates human-computer interaction by a graphical query language, which represents XML documents and DTD using a labeled XML graph. Finally, the importance of XQuery [13] is due to the fact that it is a W3C recommendation.

In Section 2 we propose the extension of GeoPQL [14] [15], called XPQL, to use on the Web. Section 3 describes a representation of the XPQL query by GML, and finally in section 4 we give conclusions and possible future developments.

2 The eXtended Pictorial Query Language

The XPQL query language is an extension of GeoPQL based on an algebra consisting of a set of spatial and temporal operators on *symbolic geographical objects* (*sgo*). Any *sgo* can also be characterized temporally. This section presents the XPQL syntax (especially with regard to its applicability between any two features) with all possible topological relations between two *sgo*, and temporal relations between *sgo* temporal attributes. It also discusses possible solutions to query interpretation ambiguities arising in particular configurations.

2.1 Base concepts

An *sgo* is formally defined as a 4-tuple $\psi = < id, objclass, \Sigma, \Lambda >$ where:

- *id* is the *sgo* identifier;
- *objclass* is the set (possibly empty) of classes iconized by ψ;

- Σ represents the attributes to which the user can assign a set of values; this allows selection to be made among the classes of objects or their instances, iconized by ψ. Some attributes can be referred to a temporal dimension. In particular, different kinds of temporal dimensions can be considered, represented by temporal intervals or instants.
- Λ is the ordered set of pairs (h, v), which defines the spatial characteristics and position of the *sgo* with respect to a reference point in the working area.

From a geometric point of view, an *sgo* is defined by its border and possible internal points. The definition, evaluation and applicability of the operators are defined in terms of these characteristics.

Let us go on to the semantics and properties of the different operators, using the following symbolism: Let ψ be an *sgo,* we have:

$\partial\psi$ = geometric border of ψ, defined as:

- if ψ is a polygon, $\partial\psi$ is the set of its accumulation points (as defined in the set theory);
- if ψ is an open polyline, $\partial\psi$ is formed by its extreme points;
- if ψ is a point ∂, ψ is the empty set.

$\psi^{o} = \psi - \partial\psi$ = geometrical interior of ψ.

Dim(ψ) = 0 if ψ contains at least one point but no polylines or polygons
1 if ψ contains at least one polyline but no polygons
2 if ψ contains at least one polygon.

The previously defined *sgo* form, together with the null element, the elements of alphabet $\mathbb{A}$. This alphabet and the operators defined in Section 2.2 form the reference model.

We refer below to the h-th attribute of the object ψ_i with $\sigma_{\psi_i}^{h}$. The attributes referred to temporal dimensions can assume temporal intervals or instants as values. In this paper an attribute can be described by only one temporal interval or instant.

2.2 The algebra of XPQL

The XPQL algebra consists of twelve spatial and seven temporal operators. Other operators can also be expressed for alphanumeric values. The spatial operators are: G-union, G-difference, G-disjunction, G-touching, G-inclusion, G-crossing, G-pass-through, G-overlapping, G-equality, G-distance, G-any and G-alias. The temporal operators are: T-before, T-meets, T-overlaps, T-starts, T-during, T-finishes and T-equals.

Below is a brief formal description of these operators.

G-union definition (Uni): The *G-union* of two *sgo* ψ_i e ψ_j is a new *sgo* defined as the set of all points belonging to ψ_i and/or ψ_j.

Formally: Let ψ_i, $\psi_j \in \mathbb{A}$ be two sgo. Then:

$$\psi_i \text{ Uni } \psi_j \equiv \psi_h = \{ x: x \in \psi_i \vee x \in \psi_j \} \text{ and } \psi_h \in \mathbb{A}.$$

G-touching definition (Tch): G-touching between two *sgo* ψ_i and ψ_j exists iff the points common to the two ψ are all contained in the union of their boundaries. If this condition is satisfied, the result of this operation between ψ_i and ψ_j is a new ψ called ψ_h and defined by the set of points common to ψ_i and ψ_j .

Formally: Let ψ_i, $\psi_j \in \mathtt{A}$ be two sgo. Then:

ψ_i Tch $\psi_j \equiv \psi_h = \{ x: x\in\psi_i \wedge x\in\psi_j \wedge x\in(\partial\psi_i \cup \partial\psi_j)\}$ and $\psi_h \in \mathtt{A}$.

G-inclusion definition (Inc): An *sgo* ψ_i G-includes another *sgo* ψ_j (and we write ψ_i Inc ψ_j) iff all the points of ψ_j are also points of ψ_i. The result is an *sgo* ψ_h which coincides with the second operand ψ_j.

Formally: Let ψ_i, $\psi_j \in \mathtt{A}$ be two sgo, and let $(\psi_i \cap \psi_j = \psi_j) \wedge (\psi_i \cap \psi_j \neq \psi_i)$. Then:

ψ_i Inc $\psi_j \equiv \psi_h = \{x : x\in\psi_j \rightarrow x\in\psi_i\}$ and $\psi_h \in \mathtt{A}$.

G-disjunction definition (Dsj): Two *sgo* ψ_i and ψ_j are G-disjoined between them (formally ψ_i Dsj ψ_j) if the intersection of their borders *AND* the intersection of their internal points is null.

Formally: Let ψ_i, $\psi_j \in$ A be two *sgo* and let

$(\partial\psi_i \cap \partial\psi_j = \emptyset) \wedge (\psi^\circ_i \cap \psi^\circ_j = \emptyset) \wedge (\partial\psi_i \cap \psi^\circ_j = \emptyset) \wedge (\psi^\circ_i \cap \partial\psi_j = \emptyset)$ Then:

ψ_i Disj $\psi_j \equiv$ *true*.

G-pass-through definition (Pth): Let ψ_i be a polyline and let ψ_j be a polygon. Then, the operator G-pass-through is applicable iff the polyline is partially inside the polygon G-disjoined between them (formally ψ_i Pth ψ_j).

Formally: Let ψ_i, $\psi_j \in \mathtt{A}$ be two sgo, where ψ_i is a polyline and ψ_j is a polygon, and let

$(\psi^\circ_i \cap \partial\psi_j \neq \emptyset) \wedge (\psi^\circ_i \cap \psi^\circ_j \neq \emptyset)$ Then:

ψ_i Pth $\psi_j \equiv \psi_h = \{x: x \in \psi_i \cap \psi_j \}$ and $\psi_h \in \mathtt{A}$.

G-distance definition (Dst): Let ψ_i, $\psi_j \in \mathtt{A}$ be two *sgo* of any type. Their distance is valuable and ≥ 0 iff their intersection is null. The (minimum) distance Dst (ϕ = min) between them is a numeric value that represents this distance. This operator can be used to find all *sgo* having distance θ (θ being one of the following symbols: $>$, $<$, $=$, $\leq$, $\geq$, $\neq$) from the reference *sgo*. The distance (δ_ϕ) value is given by: $\delta_\phi(\psi_i, \psi_j)_\theta = \psi_h$ where

- ψ_h indicates a bi-oriented segment representing the distance operator between ψ_i, ψ_j
- ϕ is the qualifier which solves this ambiguity.
- θ is a selection expression that includes conventional operators ($>$, $<$, $=$, $\neq$, etc.) or methods that behave like operators.

G-difference definition (Dif): Let ψ_i, $\psi_j \in \mathtt{A}$ be two sgo. The difference between two symbolic objects ψ_i and ψ_j is defined as a new *sgo* (ψ_h) which contains all the points which belong to ψ_i but not to ψ_j.

Formally: Let ψ_i, $\psi_j \in \mathtt{A}$ be two sgo,

ψ_i Dif $\psi_j := \psi_h = \{x : x\in\psi_i \wedge x\notin\psi_j)$ and $\psi_h \in \mathtt{A}$.

G-crossing definition (Crs): Let ψ_i, $\psi_j \in \mathtt{A}$ be two polylines, and let $\psi^\circ_i \cap \psi^\circ_j \neq \emptyset$. Then, Cross: ψ_i Crs $\psi_j := \psi_h = \{x: x \in \psi^\circ_i \cap \psi^\circ_j\}$ and $\psi_h \in \mathtt{A}$.

G-overlapping Definition (Ovl): Let ψ_i, $\psi_j \in \mathtt{A}$ be two *sgo* of the same type. A non-null overlap exists between them iff their intersection is also non-null and has the same dimension as the sgo.

Formally: Let ψ_i, $\psi_j \in \mathtt{A}$ be two *sgo* of the same type (polyline or polygon) and let $\dim(\psi_h) = \dim(\psi_i) = \dim(\psi_j) \wedge (\psi_h \neq \psi_i) \wedge (\psi_h \neq \psi_j)$. Then

ψ_i Ovl $\psi_j := \psi_h = \{x: x \in \psi_i \wedge x \in \psi_j\}$ and $\psi_h \in \mathtt{A}$

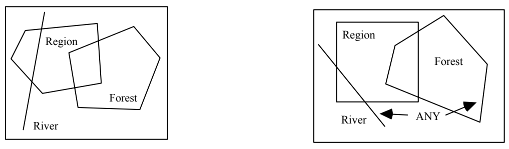
G-equality definition (Eql): Two symbolic geographical objects ψ_i and ψ_j are topologically equal if they are of the same type and have the same shape.

Formally: Let ψ_i, $\psi_j \in \mathtt{A}$ be two *sgo* of the same type and let $(\psi_i \cap \psi_j = \psi_j) \wedge (\psi_i \cap \psi_j = \psi_i)$

Then: ψ_i Eql $\psi_j := \psi_h = \{x: x \in \psi_i \wedge x \in \psi_j\}$ and $\psi_h \in \mathtt{A}$

G-any Definition (Any): Let ψ_i, $\psi_j \in$ A be two sgo. Between them any admissible relationship is valid if the G-any operator is applied between them.

Given a fixed configuration formed by two *sgo* and let $\Re$ be the set of all admissible relationships between them, we have: ψ_i Any $\psi_j := \psi_i \; \Re \; \psi_j =$ true.



(a) (b)

Fig. 1. The use of the G-any operator

To explain the use of the G-any operator, consider the following query: "Find all the regions which are *passed through* by a river and *overlap* a forest", where the user has no interest in the relationship between river and forest. If the user draws the configuration of Fig. 1-a in such a way that the forest and the river are "Disjoined", then the system must interpret the query as "Find all the regions which are *passed through* by a river and *overlap* a forest, *and the forest is disjoined from the river*". Other query representations imply a spatial relationship between the forest and the river.

By the introduction of the G-any operator, the query is correctly drawn as in Fig. 1-b, and it is possible to give the exact interpretation of the query with respect to the answer desired by the user.

G-alias Definition (Als): Let ψ_i be a sgo. ψ_j is an *alias* of ψ_i if the only difference between them is their shape.

G-alias allows implementation of a query with the OR operator (in practice, *G-alia*s duplicates the same *sgo* in order to draw a query in which this *sgo* is used in the two operands of the OR operator in this query).

For example, Fig. 2 shows the pictorial representation of the query "Find the regions which are *passed through* by a river OR which *include* a lake".

The temporal operators refer to the temporal attributes of two sgo. For example, if one *sgo* represents a province and another represents a road, we can ask to find all the roads built between the years 1980 and 1990 that pass through provinces created after 1975, or, for example, to find all roads that pass through provinces created after 1975 and were built since the province was created.

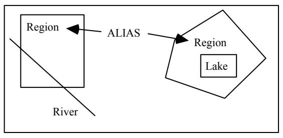


Fig. 2. a pictorial representation of a query

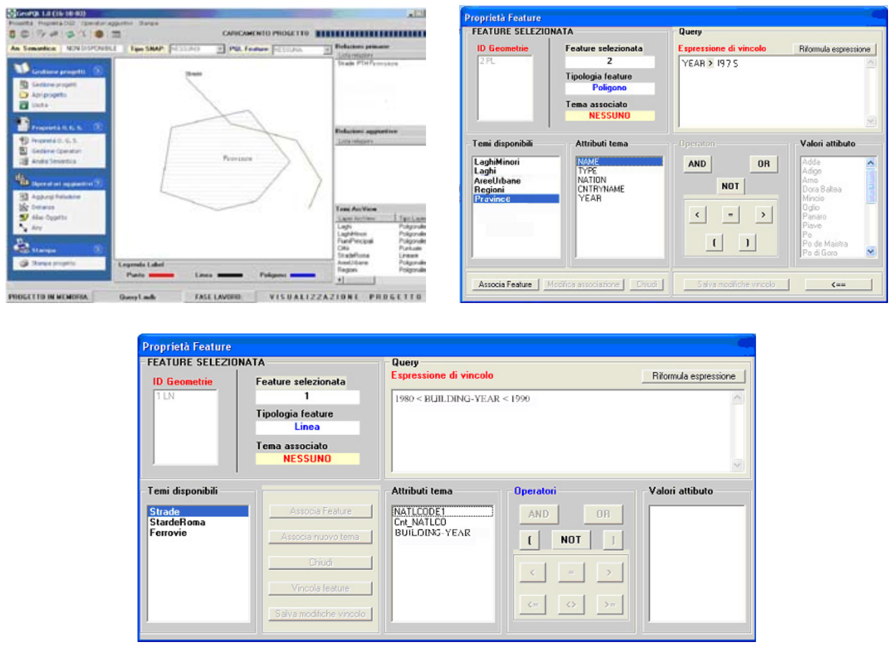
T-before (Bef) Definition: An attribute $\sigma_{\psi_i}^h$ of the *sgo* ψ_i is *T-before* another attribute $\sigma_{\psi_j}^k$ of the *sgo* ψ_j if $\sigma_{\psi_i}^h$ takes values (of the interval or instant) temporally "before" $\sigma_{\psi_j}^k$ (of the interval or instant). The attributes can take both intervals and instants as values.
T-meets (Mts) Definition: An attribute $\sigma_{\psi_i}^h$ of the *sgo* ψ_i *T-meets* another attribute $\sigma_{\psi_j}^k$ of the *sgo* ψ_j if $\sigma_{\psi_i}^h$ takes the maximum value (of the interval or instant) temporally "coincident" with the minimum value of $\sigma_{\psi_j}^k$ (of the interval or instant). The attributes can take both intervals and instants as values.
T-overlap (TOv) Definition: An attribute $\sigma_{\psi_i}^h$ of the *sgo* ψ_i *T-overlaps* another attribute $\sigma_{\psi_j}^k$ of the *sgo* ψ_j if $\sigma_{\psi_i}^h$ takes the minimum value temporally "before" the minimum value of $\sigma_{\psi_j}^k$ and the maximum value temporally "after" the minimum value of $\sigma_{\psi_j}^k$. Both attributes can take intervals as values.
T-starts (Sts) Definition: An attribute $\sigma_{\psi_i}^h$ of the *sgo* ψ_i *T-starts* another attribute $\sigma_{\psi_j}^k$ of the *sgo* ψ_j if $\sigma_{\psi_i}^h$ takes the minimum value (of the interval or instant) temporally "coincident" with the minimum value of $\sigma_{\psi_j}^k$ (of the interval or instant). The attributes can take both intervals and instant as values.
T-during (Drg) Definition: An attribute $\sigma_{\psi_i}^h$ of the *sgo* ψ_i is defined as *T-during* another attribute $\sigma_{\psi_j}^k$ of the *sgo* ψ_j if $\sigma_{\psi_i}^h$ takes the minimum value temporally "after" the minimum value of $\sigma_{\psi_j}^k$ and the maximum value temporally "before" the maximum value of $\sigma_{\psi_j}^k$. Both attributes can take intervals as values.
T-finishes (Fns) Definition: An attribute $\sigma_{\psi_i}^h$ of the *sgo* ψ_i *T-finishes* another attribute $\sigma_{\psi_j}^k$ of the *sgo* ψ_j if $\sigma_{\psi_i}^h$ takes the maximum value (of the attribute or instant) temporally "coincident" with the maximum value of $\sigma_{\psi_j}^k$ (of the interval or instant). The attributes can take both intervals and instants as values.
T-equals (Tes) Definition: An attribute $\sigma_{\psi_i}^h$ of the *sgo* ψ_i *T-equals* another attribute $\sigma_{\psi_j}^k$ of the *sgo* ψ_j if $\sigma_{\psi_i}^h$ takes the minimum and maximum values (of the interval or

instant) temporally “coincident” with the minimum and maximum values of $\sigma_{\psi_j}{}^{k}$ (of the interval or instant). The attributes can take both intervals and instants as values.

2.3 The XPQL query language

Users interact by means of the XPQL primitives to query databases in different Web servers. Specifically, each client can access a database if the user is logged in and has the correct access rights. The user selects layers of interest available from different geographical databases and draws an XPQL query. This must be translated according to GML syntax. The GML query must be sent to the logged-in servers. ArcView® accesses the geographical database (for each Web server) to obtain all the results for the expressed query.

Servers answer the query, sending the GML expression of results to the client and presenting them to the user by SVG using the correspondence defined in [16] between GML and SVG.



Proprietà Feature
FEATURE SELEZIONATA
ID Geometrie: 2 PL
Feature selezionata: 2
Tipologia feature: Poligono
Tema associato: NESSUNO
Query
Espressione di vincolo: YEAR > 1975
Temi disponibili: LaghiMinori, Laghi, AreeUrbane, Regioni, Province
Attributi tema: NAME, TYPE, NATION, CNTRYNAME, YEAR

Proprietà Feature
FEATURE SELEZIONATA
ID Geometrie: 1 LN
Feature selezionata: 1
Tipologia feature: Linea
Tema associato: NESSUNO
Query
Espressione di vincolo: 1980 < BUILDING-YEAR < 1990
Temi disponibili: Strade, StardeRoma, Ferrovie
Attributi tema: NATLCODE1, Cnt_NATLCO, BUILDING-YEAR

Fig. 3. a pictorial representation of a query with its temporal constraints

Fig. 3 shows the pictorial representation of the query: “Find all roads built between the years 1980 and 1990 that pass through provinces created after 1975”. The temporal constraints (accessible by means of sub-views) are:

Province.year > 1975

1980 < Roads.building-year < 1990

As can be seen, the user has only to be able to draw one or more points, polylines and polygons. He does not know the XPQL algebra operators, nor its syntax. If the

user query is: “Find all roads that passthrough provinces created after 1975 and which were built after the province was created”, the temporal constraints must be specified as follows:
Province.year > 1975
Roads.building-year > Province.year

3 Translation of the query from XPQL to GML

In XPQL the user formulates symbolic queries, formed by graphical objects which symbolically represent types and geographical objects. Their relevant role in the query is not for their real form, but for their feature type and the (topological) relationships existing among them. In query translation from XPQL to GML it is therefore necessary to translate both the graphical object feature types and the different relationships existing among pairs of the drawn symbolic objects. GML provides various kinds of objects for describing geography including features, reference coordinate systems, geometry, topology, time, units of measurement and generalized values.

3.1 Coding data type

The real world in the geographic domain can be represented as a set of features, and AbstractFeatureType codifies a geographic feature in GML. The state of a feature is defined by a set of properties (attribute), and each of these is described by a triple {name, type, value}. The number of properties a feature may have, together with their names and types, are determined by its type definition. Geographic features with geometry are those with properties that may be geometricallyvalued. A feature collection is a collection of features that can itself be regarded as a feature; as a consequence a feature collection has a feature type and thus may have distinct properties of its own, in addition to the features it contains.

A query XPQL (represented by a set of *sgo*) is translated into a GML feature collection. Each feature is characterized by a set of properties. Its geometry type is a very important property and is given in the reference coordinate system. This property is included in the more general property location of the AbstractFeatureType describing the extent, position or relative location of the feature.

A reference system provides measurement scale for assigning values “to a location, time or other descriptive quantity or quality”. Geometries in GML indicate the reference coordinate system in which their measurements have been made. The “parent” geometry element of a geometric complex or aggregate indicates this for its constituent geometries.

GML codifies the fundamental geometric types in CoordinatesType in geometry.xsd. The most important correspond to the following geometric classes: PointType, MultiPointType, LinearRingType, LineStringType, MultiCurveType, PolygonType, MultiSurfaceType, MultiGeometryType.

CoordinatesType is defined as a text string recording an array of tuples or coordinates. Some attributes support the description of the internal structure.

```
CoordinatesType:
  <complexType name="CoordinatesType">
    <simpleContent>
      <extension base="string">
        <attribute name="decimal" type="string" default="."/>
        <attribute name="cs" type="string" default=","/>
        <attribute name="ts" type="string" default="&#x20;"/>
      </extension>
    </simpleContent>
  </complexType>

  Element coordinates
  <element name="coordinates" type="gml:CoordinatesType" />
```

Where the *decimal* symbol is used for a decimal point (default=“.” a stop or period), the *cs* symbol is used to separate components within a tuple or coordinate string (default=“,” a comma) and the *ts* symbol is used to separate tuples or coordinate strings (default= “ ” a space)

The symbolic XPQL features represented in GML are Point, Polyline, Polygon. They are represented by the GML geometric classes PointType, LineStringType and PolygonType. A PointType is defined by a single coordinate tuple. A LineStringType is a special curve that consists of a single segment with linear interpolation. It is defined by two or more coordinate tuples, with linear interpolation between them. A PolygonType is a surface that is defined by a single (not multiple) surface. The boundary is coplanar and the polygon uses planar interpolation in its interior.

```
Complex Type PointType
  <complexType name="PointType">
    <complexContent>
      <extension base="gml:AbstractGeometricPrimitiveType">
        <sequence>
          <choice>
            <element ref="gml:pos" />
            <element ref="gml:coordinates" />
            <element ref="gml:coord" />
          </choice>
        </sequence>
      </extension>
    </complexContent>
  </complexType>

Complex Type LineStringType
  <complexType name="LineStringType">
    <complexContent>
      <extension base="gml:AbstractCurveType">
        <sequence>
          <choice>
            <choice minOccurs="2" maxOccurs="unbounded">
              <element ref="gml:pos" />
              <element ref="gml:pointRep" />
              <element ref="gml:coord" />
            </choice>
            <element ref="gml:coordinates" />
          </choice>
        </sequence>
      </extension>
    </complexContent>
```

```
   </complexType>

Complex Type PolygonType
   Complex Type PolygonType
   <complexType name="PolygonType">
    <complexContent>
      <extension base="gml:AbstractSurfaceType">
       <sequence>
        <element ref="gml:exterior" minOccurs="0" />
        <element ref="gml:interior" minOccurs="0" maxOccurs="unbounded" />
       </sequence>
      </extension>
    </complexContent>
   </complexType>
```

The remaining geometric classes do not explicitly define the *sgo* in XPQL, because a query cannot distinguish between MultiSurfaceType and PolygonType, both expressed according to the *sgo* Polygon. In fact, an XPQL query cannot distinguish between an *sgo* formed by one polygon and one involving more than one polygon. Neither can a query expressing the polygon cardinality be formulated. The *sgo* Polygon represents both i) *sgo* described by one polygon only and ii) *sgo* described by a set of two or more polygons.

The XPQL query can be expressed with a graphical approach, using a local coordinate system of the working area (window used for the query). In contrast to the visualization area of geographical data, which is constantly connected to the reference coordinate system of the involved cartography, the working area does not need to be connected because its constraints do not involve geographical coordinates. For example, a user can express, the following constraints independently of geographical coordinates: i) a river passes through a region, ii) a road crosses another one. Translation of the XPQL query expression into GML constraints is usually independent of the reference coordinate system. Such constraints refer explicitly to the geometric topology and relations between the involved features.

Sometimes, however, the user may need to express constraints explicitly referring to the geographical coordinates (e.g. the crossing of two roads near a given location). For this reason XPQL enables users to explicitly express constraints referring to the reference coordinate system. This is represented in GML through locations, using geometry in the reference coordinate system.

The location property describes the general location of the feature. In the instance document representing an XPQL query, an element of this type might appear as follows:

```
   <gml:location>
     <gml:Point gml:id="crossroad6" srsName="crsg:1126">
       <gml:coordinates>51:43:00N 45:50:00W</gml:coordinates>
     </gml:Point>
   </gml:location>
```

XPQL considers Instant and Interval as primitives for temporal information. GML defines them as the geometric primitives in temporal dimension. Instant is a zero-dimensional primitive representing the time position.

The element gml:TimeInstant is:

```
<element name="TimeInstant" type="gml:TimeInstantType"
    substitutionGroup= "gml:_TimePrimitive"/>
```

```
<complexType name="TimeInstantType" final="#all">
  <complexContent>
    <extension base="gml:TimePrimitiveType">
      <sequence>
        <element ref="gml:timePosition"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

TimePosition can be connected to the temporal reference system, and a different precision information level is specified considering *year, year-month, date, or dateTime* in the Element *TemporalPositionType,* as follows:

```
<simpleType name="TemporalPositionType">
  <union memberTypes="dateTime date gYearMonth gYear anyURI decimal"/>
</simpleType>
```

A temporal value is associated to a temporal reference system using the frame attribute. The calendar era name to which a precise date is referred is given by calendarEraName, while a not well defined temporal position can be expressed by indeterminatePosition, as follows:

```
<complexType name="TimePositionType" final="#all">
  <simpleContent>
    <extension base="gml:TemporalPositionType">
      <attribute name="indeterminatePosition"
        type="gml:TimeIndeterminateValueType" use="optional"/>
      <attribute name="calendarEraName" type="string" use="optional"/>
      <attribute name="frame" type="anyURI" use="optional" default="#ISO-
        8601"/>
    </extension>
  </simpleContent>
</complexType>
```

Now we consider the gml definition of Interval. It is defined by gml:TimePeriod, gml:_duration The element gml:TimePeriod is:

```
<element name="TimePeriod" type="gml:TimePeriodType"
substitutionGroup="gml:_TimePrimitive"/>
<complexType name="TimePeriodType" final="#all">
  <complexContent>
    <extension base="gml:TimePrimitiveType">
      <sequence>
        <element ref="gml:begin"/>
        <element ref="gml:end"/>
        <element ref="gml:_duration" minOccurs="0"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>

<element name="begin" type="gml:TimeInstantPropertyType"/>
<element name="end" type="gml:TimeInstantPropertyType"/>
```

The Duration is:

```
<element name="_duration" type="gml:TimeDurationType" abstract="true">
<simpleType name="TimeDurationType">
  <union memberTypes="duration decimal"/>
</simpleType>
```

3.2 Coding spatial and temporal operators

XPQL considers both spatial and temporal operators (see section 2). Operators represent properties existing between symbolic graphical objects of the query. The following schema shows a basic pattern that supports the encoding of property elements in GML 3.0 with a complex content among objects:

```
<complexType name="AssociationType">
  <sequence>
    <element ref="gml:_Object" minOccurs="0"/>
  </sequence>
  <attributeGroup ref="gml:AssociationAttributeGroup"/>
</complexType>

<element name="_association" type="gml:AssociationType"
abstract="true"/>
```

The XPQL temporal operators are derived from GML: AssociationType In addition, for the XPQL spatial operators, FeaturePropertyType is a particular class of properties (using the gml:AssociationType pattern) which defines associations between features.

```
<complexType name="FeaturePropertyType">
   <sequence>
      <element ref="gml:_Feature" minOccurs="0"/>
   </sequence>
   <attributeGroup ref="gml:AssociationAttributeGroup"/>
</complexType>
```

Thus, in GML3 the type for a property element may be derived from GML: FeaturePropertyType. The previous schema is able to support the translation of the various XPQL operator types, with the exception of G-distance. The representation of the different operators in GML3 considers restrictions relating to their applicability to the type of sgo. Such restrictions are specified in the XPQL operator definitions in Section 2. Finally, XPQL allows expression of constraints, which refer to the G-Distance operator between symbolic geographical objects. The GML MeasureType is used to represent these constraint types.

A MeasureType is an amount encoded as a double, together with a unit of measurement (UOM) indicated by a UOM attribute. The UOM value is a reference to a Reference System, either a ratio or interval scale.

```
<complexType name="MeasureType">
  <simpleContent>
    <extension base="double">
      <attribute name="uom" type="anyURI" use="required"/>
    </extension>
  </simpleContent>
</complexType>
```

The pictorial query of Fig. 3 is schematically shown in Fig. 4 in its main components using GML: an element of type province, an element of type road, an element of type G-pass-through and three elements (represented just one time) of type T-before. The Figure also shows the definitions in GML of types and operators involved in the query.

```
<!-- ==============================
type definitions for operators
========================== -->
…………
<complexType name="G-pass-throughType">
  <complexContent>
    <extension base="gml:AssociationType">
      <sequence>
        ……………………..
      </sequence>
    </extension >
  </complexContent>
</complexType>
…………
<complexType name="T-beforeType">
  <complexContent>
    <extension
    base="gml:FeaturePropertyType">
      <sequence>
        ……………………..
      </sequence>
    </extension >
  </complexContent>
</complexType>
```

```
<!-- ==============================
type definitions for different geographical layers
================================-->
…………
<complexType name="ProvinceType">
  <complexContent>
    <extension base="gml:PolygonType">
      <sequence>
        ……………………
      </sequence>
    </extension>
  </complexContent>
</complexType>
…………
<complexType name="RoadType">
  <complexContent>
    <extension base="gml:LineStringType">
      <sequence>
      …………………………
      </sequence>
    </extension>
  </complexContent>
</complexType>
…………
```

```
<!-- ============================
the query example
============================== -->

<QueryModel …………………>
<gml:name>ExampleFigure3</gml:name>
…………………..
<queryMember>
<Province>
………………..
</Province>
</queryMember>

<queryMember>
<Road>
```

```
</Road>
</queryMember>

<queryMember>
<G-pass-through>
………………..
</G-pass-through >
</queryMember>

<queryMember>
<T-before>
………………..
</T-before>
</queryMember>
………………..
</QueryModel>
```

Fig. 4. the GML representation of the pictorial query of Figure 3

4 Conclusions

The diffusion of the Web and its applications has produced particular interest in querying and accessing distributed geographical databases.

This paper presents XPQL, a Pictorial Query Language, which enables the user to easily formulate a query drawing geometrical features to conceptually represent entities of the geographical world. We start from the consideration of geographical information from a spatial point of view according to a two dimensional description and taking into account of temporal information. The XPQL primitives, and consequently each XPQL query, are translated into GML expressions in order to guarantee the exchange of and access to geographical information among different databases available to each user. In particular, geo-point, geo-polyline and geo-region can be represented using the GML types: PointType, LineStringType and PolygonType. Instant and Interval are given by TimeInstant and TimePeriod.

We are now focusing our activities on designing and implementing a new prototype with a simpler, more effective interaction approach through the growth in multimodal interaction tools. We are also studying the possibility of using this approach to formulate queries for many types of application areas which are not purely geographical in nature. Our hope is to be able to specify relational queries in this way, without the end user needing to understand the intricacies of boolean logic or set theory.

References

1. S. Cox , P. Daisey, R. Lake, C. Portele, A. Whiteside. "Geography Markup Language (GML) Implementation Specification" OGC 02-023r4, Open GIS Consortium, Inc., 2003.
2. D. Papadias, T. Sellis "A Pictorial Query-by-Example Language" Journal of Visual Languages and Computing, Vol.6, N.1, pp. 53-72, 1995.
3. S. Kaushik, E.A. Rundensteiner "SVIQUEL: A Spatial Visual Query and Exploration Language" 9th Intern. Conf. on Database and Expert Systems Applications - DEXA'98, LNCS N. 1460, pp. 290-299, 1998.
4. Calcinelli D., Mainguenaud M. "Cigales, a visual language for geographic information system: the user interface" Journal of Visual Languages and Computing, Vol. 5, N. 2, pp. 113-132, June 1994.
5. Aufaures-Portier M.A., Bonhomme C. "A High Level Language for Spatial Data Management" Third Int. Conf. on Visual Information Systems - VISUAL '99, LNCS N. 1614, pp. 325-332, 1999.
6. Meyer B. "Beyond Icons: Towards New Metaphors for Visual Query Languages for Spatial Information Systems" First Intern. Workshop on Interfaces in Database Systems, Springer-Verlag Publ., pp.113-135, 1993.
7. Egenhofer M.J. "Query Processing in Spatial-Query-by-Sketch" Journal of Visual Languages and Computing, Vol. 8, N. 4, pp. 403-424, 1997.
9. Bonifati, S. Ceri "Comparative Analysis of Five XML Query Languages", *ACM SIGMOD Record*, 29 (1), 2000.
10. D. Lee, W. W. Chu "Comparative Analysis of Six XML Schema Languages", *ACM SIGMOD Record*, 29 (3), 2000.
11. J. Robie, J. Lapp, and D. Schach. "Xml query language (Xql)". In Query Languages 1998.
12. S. Ceri, S. Comai, E. Damiani, P. Fraternali, S. Paraboschi, and L. Tanca. "XML-GL: a graphical language for querying and restructuring XML documents" *Computer Networks*, 31 (11–16), pp. 1171–1187, May 1999.
13. S. Boag, M. F. Fernandez, D. Florescu, J. Robie, J. Siméon "XQuery 1.0: An XML Query Language", W3C Working Draft 02 May 2003, http://www.w3.org/TR/xquery/ .
14. F. Ferri, F. Massari, M. Rafanelli. "A Pictorial Query Language for Geographic Features in an Object-Oriented Environment". Journal of Visual Languages and Computing, Vol. 10, N. 6, pp. 641-671, Dec. 1999.
15. F.Ferri, P.Grifoni, M.Rafanelli. "GeoPQL: a Geographical Pictorial Query Language" Tech. Rep. 2004 IASI-CNR.
16. Taladoire Gilles, "Geospatial data integration and visualisation using open standards" 7th EC-GI&GIS Workshop EGII Managing the Mosaic", Postdam, Germany, 13-15 June 2001.