--

--

SPATIAL REASONING FOR GEOGRAPHIC INFORMATION SYSTEMS

F.Coenen, B. Beattie, T.J.M.Bench-Capon, B.M.Diaz and M.J.R.Shave

Department of Computer Science,

The University of Liverpool,

Chadwick Building, P.O. Box 147,

Liverpool L69 3BX, England.

Tel: 0151 794 3698

Fax: 0151 794 3715

email: frans@csc.liv.ac.uk

ABSTRACT

Geographic Information Systems (GIS) are well established and used in many areas of application. However, they do not readily support spatio-temporal reasoning with respect to the data objects that they model. This is due to incompatibilities between the (raster/vector) representations used by current GIS and that supported by existing spatio-temporal reasoning mechanisms. In this paper we describe a mechanism whereby multi-dimensional reasoning capabilities can be attached to GIS. More specifically a generic representation is described that is fully compatible with the functionality of current GIS while at the same time supporting a spatio-temporal reasoning mechanism based on a constraint satisfaction paradigm. The representation has been incorporated into a demonstration spatio-temporal reasoning system which has been applied to a number of GIS applications and related areas.

1. INTRODUCTION

The work described here is motivated through the observation that current Geographic Information Systems (GIS) do not make full use of the data that they model, i.e. they do not provide for a spatio-temporal reasoning capability that can be applied to that data. However, there are many applications where such a capability would be useful. Two application areas of particular interest to the work presented here are:

1) Environmental Impact Assessment (EIA)
2) Enhanced electronic chart interaction for marine navigation

More specifically the work described provides support for applications which require the solution of N-dimensional GIS queries where information concerning the spatial objects under consideration is not directly encoded into the GIS, and/or where selections between alternative candidate solutions are required. In other words, the solution of GIS queries which involve some degree of reasoning. It is considered that provision of such a reasoning capability will increase the potential functionality of GIS.

Current GIS are based on either a raster or vector representation. To manipulate such data structures directly requires recourse to computationally complex (and cumbersome) routines. This means that these representations are unsuited to the degree of reasoning/manipulation required for many applications. It is therefore suggested that some intermediate higher-level representation is adopted which, while interfacing to both raster and vector representations, can be fully integrated with a suitable reasoning mechanism. Additional requirements for such a representation are that it is generally applicable to N-dimensional space, and that it supports computationally efficient translation and rotation of spatial objects so as to support the necessary degree of object manipulation. As the representation is not intended to be directly end-user readable, ease of understanding is not considered to be an issue.

Broadly, spatial reasoning is concerned with the manipulation of the N-dimensional relationships that exist between objects so as to reach some application-dependent conclusion. Traditionally the objective is to determine the relationships that exist between two or more specific spatial objects given a set of known relationships that link those objects to each other and to further objects. This was the principal concern of much early work on temporal reasoning (Allen 1983), and continues to be the principal aim of more recent work in spatial reasoning (Egenhofer 1994). The research discussed here extends this objective in that the aim is to determine the relationships that exist between two or more spatial objects given a set of relationships that are **desired** to exist between such objects. Further, this goal is seen as only an intermediate goal; the ultimate goal is to arrive at locations to be associated with objects specified in GIS queries. The nature of these locations will of course be dictated by the nature of the derived relationships.

Many established spatio-temporal reasoning systems are based on formal logic languages, for example Kowalski's *event calculus* (Kowalski and Sergot 1986). For mainstream spatial AI applications this is seen as providing the necessary expressive power and deductive machinery needed. Given the large amount of data to be considered in GIS applications it is suggested that mechanisms to apply such formal logics to GIS data, regardless of whether an intermediate representation is used or not, will be inefficient with respect to the data store, and consequently unlikely to operate in a reasonable time. Thus any proposed intermediate representation must also facilitate some other approach to expressing the relationships that are required to exist between GIS objects which can be interfaced to the necessary deductive machinery. An alternative approach to addressing this problem is to develop higher order logics which are more readily appropriate to spatial reasoning applications involving significant amounts of data (Cohn 1995). However such logics tend to be qualitative while, with respect to many GIS applications, the result must be quantifiable.

It is proposed that the required intermediate representation be founded on the theory of point set topology (Spanier 1966). This will allow for the suitable representation of location information associated with GIS objects while at the same time providing the required expressive power to define the complex relationships that exist between those objects. The use of point set topology with respect to spatial reasoning is well established, for example Egenhofer's *9-Intersection* technique (Egenhofer 1994).

A system whereby GIS queries, comprising objects and relations, are expressed in terms of sets of points and functions applied to those sets of points lends itself to solution using constraint satisfaction techniques (van Hentenryck 1989). More formally such GIS queries can be considered to comprise a set of variables (GIS objects) which can take values from some global point set (the possible locations for the objects), and a set of constraints (the desired relationships) which specify which values are compatible with each other.

With respect to the above the authors have developed and investigated a number of representation techniques founded on point set topology and incorporating the tessellation of space (i.e. the sub-division of space into isohedral sub-spaces). These representations have been incorporated into a series of demonstration spatio-temporal reasoning systems collectively referred to as the SPARTA (SPAtial Reasoning using Tesseral Addressing) systems. The current representation, which is a development from earlier representations, is described in Section 2. This representation can then be used to describe objects contained in an *object space* (Section 3), and the relationships that are required to exist between those objects. These relations are defined in terms of a set of reasoning primitives as described in Section 4. These primitives can then be interpreted as constraints, the satisfaction of which is considered in Section 5. Some discussion concerning user interaction and usage is then presented in Section 6, and an overview of the appropriate deductive machinery in Section 7. Some conclusions are presented in Section 8.

2. REPRESENTATION

The present version of the proposed intermediate representation is founded on the standard Cartesian representation of 4-D space on the grounds that this is the most popular, and consequently the most readily accessible method used to present 4-D data. However, to achieve storage advantages, coordinates associated with elements making up a 4-D space are compressed into a single integer referred to as an *address*. In the current version of the SPARTA system the desired compression is based on the allocation of groups of bits to individual coordinates as follows:

```
S 4444444 33333333 22222222 11111111
```

Where `S` is the sign bit, and the integers `1`, `2`, `3` and `4` represent the bits to be associated with particular coordinates used to reference 4-D space. Note that:

1) The numbering has been used to indicate the relative position of each group of bits with respect to the least significant bit. The numbering does not necessarily imply that any dimensions should be associated with any particular grouping.
2) Any other distribution of bits could equally well have been allocated provided that each dimension of interest comprises a continuous sequence of bits (we do not have to include groupings for dimensions which are not relevant to a

particular application).

3) It is not necessary to allocate identical numbers of bits to each dimension.

In order to identify the coordinate groupings a *base* must be allocated to each group. In the above case the following is appropriate:

```
BASE1 = 1
BASE2 = 256
BASE3 = 65536
BASE4 = 16777216
```

Each base represents the integer that must be applied to a particular coordinate so that it is encoded using the appropriate grouping of bits. With respect to this document the dimensions to be associated with the `BASE1`, `BASE2`, `BASE3`, `BASE4` groupings are labelled as `X`, `Y`, `Z` and `T`, although any other labelling could equally well have been used. The important point to note is that any implications that might be associated with the labelling are not relevant to the operation or usage of the system. Given the above, addresses can be calculated using the identity:

```
Address = X + BASE1(Y) + BASE2(Z) + BASE4(T)
```

An important feature of this addressing system is that it has the effect of linearising space. This linearisation can be traced in the fragment of a 2-D encoding presented in Figure 1.

This linearisation has a number of important (and desirable) consequences:

1) It is analogous with raster GIS representations and thus provides for compatibility with such representations (and consequently vector representations).
2) It allows any N-dimensional space to be treated in one-dimensional terms.
3) It supports run-line encoding.
4) It provides for convenient comparison of sequences of addresses, for example to determine if one sequence is wholly contained in another it is only necessary to compare the start and end addresses for each sequence.
5) Unlike other addressing systems, such as the quad-tesseral system (Diaz and Bell 1986), it is predictable, i.e. the relative location of the next address in the sequence, with respect to any current address, is always known (except at the extreme edges of the system). This offers significant computational advantages when (say) determining the points contained in a box.

A further advantage of the representation is the ease with which translation can be achieved. Each address can also be considered to represents a vector quantity - the direction and distance measured from the origin (the 0 address) to the indicated location. Consequently translation can be achieved simply through "vector" addition (or subtraction) by adding appropriate addresses. Thus, in Figure 1, if we wish to translate two tiles to the right and three tiles upwards from any address, the vector quantity (address) 770 should be added to the address in question. This feature can be explained if it is acknowledged that addresses of the above form are complex numbers (Jeffrey 1987), where `X` represents the real part, and `BASE1(Y) + BASE2(Z) + BASE4(T)` the imaginary part. Thus translation can be achieved through simple and

+Y

1020	1021	1022	1023	1024	1025	1026	1027	1028
764	765	766	767	768	769	770	771	772
508	509	510	511	512	513	514	515	516
252	253	254	255	256	257	258	259	260
−4	−3	−2	−1	0	1	2	3	4
−260	−259	−258	−257	−256	−255	−254	−253	−252
−516	−515	−514	−513	−512	−511	−510	−509	−508
−772	−771	−770	−769	−768	−767	−765	−764	−763
−1028	−1027	−1026	−1025	−1024	−1023	−1022	−1021	−1020

−X   +X

−Y

Figure 1: *Fragment of addressing mechanism applied to 2-D space*

efficient integer addition (and subtraction).

Rotation can also be achieved in a manner analogous to complex number theory.

3. OBJECT SPACE

Using the above intermediate representation any N-dimensional space can be defined in terms of a **positive** set of addresses (points). The resulting space described is referred to as the *object space* - the space in which all (GIS) objects of interest are considered to exist. It is necessary to include negative addresses in the representation so that, where appropriate, translation using negative vector quantities is supported.

4. REASONING PRIMITIVES

So that GIS reasoning queries can be handled, two classes of GIS object are identified, *fixed* objects and *free* objects, of which the second is further divided into two sub-classes, *shaped* and *shapeless*. The distinction between these classes is in the nature of the information available concerning individual instances of each class and, as a result, the reasoning primitives that can be applied to specific classes of object. In the following four sub-sections each of the above classes and sub-classes will be considered in turn.

4.1. Fixed Objects

Fixed objects are those GIS objects whose location is "hard coded" into a GIS (i.e. objects whose location in N-dimensional space is precisely known). Note that this location information also implies object shape and size, and attributes such as whether a location can in some sense be considered to be contiguous or not. The only reasoning that can be applied to fixed objects is in the form of filters expressed in terms of predicates that return "true" or "false" according to whether certain relations exist (or do not exist) between pairs of such objects. For example:

```
X isSubsetOf Y
```

where X and Y are sets of addresses representing the locations of two fixed objects. This filter thus tests whether the location of X is wholly contained in Y; if so the locations associated with the two objects in question are considered to be valid locations.

Despite the simplicity of such filters, they have been successfully used to verify and validate timetables where the tables are presented as two dimensional "maps" (Coenen et al. 1995).

4.2. Free Objects

Free objects usually form the subject of the GIS queries that the mechanism described is intended to support, i.e. free objects are GIS entities whose location(s), within the domain of a GIS (the object space), are to be established. The system supports two types of free object - shaped and shapeless objects.

4.2.1. Shaped Free Objects

Shaped free objects take a predefined shape and a *location space* in which the object is considered to exist (by default this location space may extend to the entire domain represented by a particular GIS application). The shape is described in terms of a set of addresses (founded on the origin of the addressing system - the zero tile).

Shape definitions may be labelled as being either *fixed* or *meta* definitions, where the latter defines a maximal object shape. Meta shape definitions are thus appropriate where the shape of an object is only partially known in that its final shape is dependent on the location of other objects.

Given a shaped object a number of *candidate* locations for the object can be determined by identifying the number of ways in which the shape can be "fitted" into the prescribed location space. Where only one candidate location is produced the object in question can effectively be considered to be a fixed object. Where no candidate locations can be found the object cannot be physically realised and hence no solution can be produced for the query under consideration.

With respect to meta shape definitions, current versions of the system also permit specification of requirements that: (1) each generated candidate location should be "clipped" with respect to other locations, (2) candidate locations must be of a minimum size if they are to be considered valid, and (3) sets of addresses representing candidate locations are contiguous. These features provide the means whereby the system can operate with objects whose shape is only partially known.

The reasoning primitives that can be used with shaped objects comprise the *filtering* of sets of candidate locations to produce revised sets of such locations. This filtering can be implemented with respect to other free shaped objects or fixed objects. For example given a set of candidate locations $\{X_1, X_2, X_3\}$ and a fixed location `Y`, a filter of the following form may be expressed:

$$\{X_1, X_2, X_3\} \text{ is NotSubsetOf } \{Y\}$$

(any candidate location `X` is a valid location if it does not overlap the location `Y`). Alternatively, given two sets of candidate locations a filter of the form:

$$\{X_1, X_2, X_3\} \text{ is NotSubsetOf } \{Y_1, Y_2, Y_3\}$$

may be expressed (any candidate location X is a valid location if it does not overlap any candidate location Y). In the first case only one compatible set of solutions will be produced. In the second case a number of solutions may result limited by the size of the Cartesian product of the two sets of candidate addresses.

The style of reasoning supported by filters of the above form is exemplified by classic AI "puzzles" such as the "N-queens" problem. In GIS terms this provides the means to address queries to find the most appropriate site for installations that have a predefined shape, for example runways, sports fields, etc.

4.2.2. Shapeless Free Objects

A shapeless free object is a GIS entity which has no fixed location and no (known) shape. In some cases all that may be known about such objects is that they exist somewhere in the domain covered by a particular GIS application. In such cases the objects have a *location space* associated with them (in which they can be said to exist) that extends to the entire domain encompassed by the application. In other cases this location space may be limited to a sub-set of the domain. In addition, for some applications, further information may be available such as a minimum size value and/or a requirement that the set of addresses describing the location space be contiguous.

The reasoning that can be associated with shapeless free objects comprises various *mapping* operations that can be applied to location spaces so as to redefine those spaces. Such operations may be defined with respect to either fixed or shaped free objects. Effectively, such mappings serve to "prune" the location spaces. Experiments using this form of reasoning have been run to determine the nature of noise pollution envelopes surrounding road systems (Beattie et al. 1996). The mechanism could equally well be applied to the identification of (say) appropriate construction sites for housing projects or industry units.

5. SPATIAL RELATIONSHIPS/CONSTRAINTS

From the above we can identify a number of reasoning primitives that can be applied to fixed and free objects. Broadly these can be categorised as being either:

1) Filters which are used to test the efficacy of locations (candidate or fixed) associated with particular objects.
2) Mappings which are used to prune location spaces associated with shapeless free objects.

These primitives form the basis for the definition of multi-dimensional relationships expressed in the form of constraints. This in turn allows for the definition of complex compound queries concerning GIS entities. In current versions of the system such constraints are expressed in terms of an operator and two operands. Currently five filter operations are supported, *isSubsetOf*, *isNotSubsetOf*, *isEqualTo* *isNotEqualTo* and *isIntersectionOf*; and two mapping operations, *intersection* and *complement*. In the simplest case the operands represent location data associated with particular objects. If this object is a fixed object the operand will comprise only a single set of addresses describing the fixed location of the object. In the event of a shaped object the operand will comprise a list of sets of addresses each describing a candidate location for the object. Where a shapeless object is under consideration the operand will again comprise a single set of addresses, but on this occasion describing the location space associated with the object.

The solution of such constraints return one or more pairs of lists of sets of addresses, where each solution pair describes a group of compatible locations for the objects in question according to the nature of the filter/mapping. The number of solution pairs returned will depend on the nature of the objects represented by the operand. Where one of the objects is a fixed object only one solution pair will be returned. Where the constraint links a shaped object to a shapeless object the number of solution pairs will be equivalent to the number of candidate solutions for the shaped object. In the event of a relation linking two shaped objects the maximum number of solution pairs will be equivalent to the size of the Cartesian product of the two groups of candidate solutions associated with each object.

Where the operands directly represent location data, using these filters and mappings, only topological relations can be expressed. For example all candidate locations associated with a shaped object that overlaps (*isIntersectionOf*) with the location associated with a fixed object can be filtered out. Alternatively a location space associated with a shapeless object can be caused to be contained (*intersection*) within a location associated with a fixed object by mapping an intersection operation onto the locations in question. To increase the expressiveness of these relations the system supports a mechanism whereby operands can be generated by applying offsets to locations. As a result filters can be expressed that return only those candidate locations that are (say) to the "northwest" of some other location. Alternatively, using offsets, a mapping can be expressed to prune a location space so that it only encompasses some space to (say) the "northwest" of some other space. Two styles of offset can be identified *global* offsets and *local* offsets. The distinction is that the first is applied to all the location addresses associated with an object, while the second is only applied to a reference address (the address nearest to the origin of the representation which is contained in the minimum bounding box surrounding the location set of addresses).

To further increase the expressiveness of the relations, provision is also made to specify operands that can be clipped with respect to fixed objects, and/or are contiguous.

In the above manner any space (constraint operand) can be defined with respect the location(s) associated with a particular object.

6. USER INTERACTION

Using the suggested representation complex compound queries can be expressed involving many objects, and relations that link these objects. Such queries are referred to as scripts and, with respect to current versions of the SPARTA system, are expressed using a scripting language. This uses a PROLOG (Bratko 1986) style syntax and provides for the definition of objects and relations in the form of *facts*. Three types of fact are supported, classes, instances and constraints. The first allows the definition of classes of GIS objects, the second the declaration of specific instances of classes, and the third the relationships (constraints) that we wish to exist between instances (or groups of instances). The scripting language incorporates many facilities to provide support for, and simplify, the generation of scripts and is currently undergoing further development.

Completed scripts are presented to the SPARTA system and, on successful parsing, translated into the proposed intermediate representation. The constraint satisfaction mechanism (see below) is then invoked. Experiments to date have shown that simple scripts are relatively easy to define. If necessary addresses can always be expressed in terms of Cartesian coordinates and converted to the appropriate form. Eventually it is anticipated that a "point and click" style interface will be developed which will serve to limit substantially any risk of input error and provide a much more intuitive interface.

7. OPERATION

In the introduction to this paper it was observed that spatial queries couched in terms of the proposed intermediate representation can more generally be defined as Constraint Satisfaction Problems (CSP). There are two broad techniques that can be used to solve CSP. One approach is to generate all possible assignments of values to variables and test these against the given set of constraints, in a manner that will result in early identification of fruitless assignments (see van Hentenryck (1989) for further discussion). The other is to use constraint propagation techniques whereby variables range over a set of values and the constraints are used to prune this set of values (see Mackworth and Freuder (1985) for further discussion). Using the intermediate representation suggested here either approach, or any specific technique encompassed by these approaches, can be adopted. However, for all implementations of the SPARTA system to date the latter has been used.

In current implementations of the SPARTA system partial and end solutions are stored in a solution tree, each node of which represents a solution state after a number of constraints have been satisfied. Branches in the tree indicate nodes where, on satisfaction of a constraint, more than one solution results. This tree is constructed dynamically as the solution process progresses. There are thus two efficiency concerns here:

1) To limit the growth of the solution tree, i.e. to minimise the number of nodes in the tree.
2) To limit the processing required to arrive at a complete set of solutions, i.e. to identify fruitless branches as early in the solution process as possible.

These two concerns are linked in that the early identification of fruitless branches of the solution tree will also have the effect of limiting the growth of the tree. To achieve the above the following have been adopted:

1) Commence the satisfaction process by considering only constraints involving at least one fixed object as these can only result in one possible solution (i.e. branch in the tree).
2) Once all constraints involving fixed objects have been processed select remaining constraints so that, on each iteration, satisfaction of the selected constraint will result in the minimum possible number of alternative solutions.
3) Where a number of possible solutions are produced, attempt to merge compatible solutions so as to reduce the overall number.

8. CONCLUSIONS

A mechanism whereby complex compound GIS queries can be expressed and resolved, founded on an intermediate representation, has been described. The representation is fully compatible with existing GIS representations while at the same time facilitating the application of appropriate deductive machinery to determine solutions to GIS reasoning queries. These features can be attributed to the nature of the linearisation offered by the representation, and the computationally efficient translation mechanism which it supports.

A series of demonstration systems have been developed (the SPARTA systems) which have been applied to a number GIS applications (Beattie et al. 1995), particularly environmental impact assessment applications (Beattie et al. 1996). It is also suggested that the technique described is applicable to a much wider range of spatio-temporal applications than those encompassed by GIS. Experiments have been conducted to investigate its applicability to temporal reasoning (Coenen et al. 1994), timetabling (Coenen et al. 1995), and (currently) its use in connection with the tabular verification and validation of Expert System rule-bases.

Other recent work has focused on the use of the proposed intermediate representation as a primary representation for GIS. This work has indicated that the representation will fully support the traditional functionality of such systems. The authors are currently considering a number of genuine application areas to which the SPARTA system can be applied.

REFERENCES

1) Allen, J.F. 1983. Maintaining Knowledge About Temporal Intervals. Communications of the ACM, Vol 26, No 11, pp832-843.

2) Beattie, B.J.H., Coenen, F.P., Bench-Capon, T.J.M., Diaz, B.M. and Shave, M.J.R 1995. Spatial Reasoning for GIS Using a Tesseral Data Representation. In N. Revell and A.M. Tjoa (eds.), Database and Expert Systems Applications, (Proceedings DEXA'95), Lecture Notes in Computer Science 978, Springer Verlag, pp207-216.

3) Beattie, B.J.H., Coenen, F.P., Hough, A., Bench-Capon, T.J.M., Diaz, B.M. and Shave, M.J.R. 1996. Spatial Reasoning for Environmental Impact Assessment. Third International Conference on GIS and Environmental Modelling, Santa Barbara: National Centre for Geographic Information and Analysis, WWW and CD.

4) Bratko, I. 1986. PROLOG Programming for Artificial Intelligence. Addison-Wesley, Wokingham, England.

5) Coenen, F.P., Beattie, B.J.H., Diaz, B., Bench-Capon, T.J.M. and Shave, M.J.R. 1994. A Temporal Calculus for GIS Using Tesseral Addressing. In Bramer, M.A. and Macintosh, A.L. (Eds), Research and Development in Expert Systems XI, Proceedings of ES'94, pp261-273.

6) Coenen, F.P., Beattie, B.J.H., Bench-Capon, T.J.M., Shave, M.J.R and Diaz, B.M. 1995. Spatial Reasoning for Timetabling: The TIMETABLER system. Proceedings of the 1st International Conference on the Practice and Theory of Automated Timetabling (ICPTAT'95), Napier University, Edinburgh, pp57-68.

7) Cohn, A.G., Gooday, J.M., Bennet, B and Gotts, N.M. 1995. A Logical Approach to Representing and Reasoning About Space. Artificial Intelligence Review, Vol 9, pp255-259.

8) Diaz, B.M. and Bell, S.B.M. 1986. Spatial Data Processing Using Tesseral Methods. Natural Environment Research Council publication, Swindon, England.

9) Egenhofer, M.J. 1994. Deriving the Composition of Binary Topological Relations. Journal of Visual Languages and Computing (1994) 5, pp133-149.

10) van Hentenryck, P. 1989. Constraint Satisfaction in Logic Programming. MIT Press, Cambridge, Massachusetts.

11) Jeffrey, A. 1987. Mathematics for Engineers and Scientists, 3rd Edition. Van Nostrand Reinhold, Wokingham, England.

12) Kowalski, R. and Sergot, M. 1986. A Logic-Based Calculus of Events. New Generation Computing, Vol 4, p67-95.

13) Mackworth, A.K. and Freuder, E.C. 1985. The Complexity of Some Polynomial Network Consistency Algorithms for Constraint Satisfaction Problems. Artificial Intelligence, Vol 25, pp65-74.

14) Spanier, E.H. 1966. Algebraic Topology. McGraw-Hill, New York.