14. Abstract Data Structures for GIS applications in archaeology

Clive Ruggles
School of Archaeological Studies, Department of Computing Studies and Midlands Regional Research Laboratory, University of Leicester, Leicester LE1 7RH, U.K.

14.1 Introduction: modelling with abstract objects

Conceptual models are constructed of abstractions of objects in the real world. By precisely specifying the structure of these objects, and the functional relationships between them, we can use mathematics to investigate the nature and properties of the model as a whole. Precise specification also opens up the possibility of implementing the model on a computer. This can have considerable benefits, such as an enhanced ability to perform complex manipulations upon it and the potential to visualise the results in a useful way.

Central to any attempt to implement abstractions of real-world objects on a computer is the concept of an Abstract Data Type (ADT).[1] An ADT is essentially an abstract model of a real-world object expressed through its functionality. For example, the definition of an ADT *List* might specify operations such as *Create*, which returns an empty list, *IsEmpty*, which takes a list and returns a Boolean value which is true if and only if the list is empty, *Head*, which takes a non-empty list and returns its first element, *Tail*, which takes a non-empty list and returns the list with its first element removed, and *Append*, which takes an element and a list and returns the list with the given element appended at the front. Using these operations, we could manufacture any list, investigate the properties of any list, and manipulate any list, without necessarily knowing how to represent a list (for example as an ordered sequence of elements). Equivalently, in the computer context, we can use a given *List* ADT to create and manipulate lists without knowing how the ADT is actually implemented in the computer system being used. Different systems may actually implement *List* ADT in different ways, but as long as the specification is the same in each case, the properties 'seen' by a user of each system will be identical. A familiar example of a low-level ADT is *Integer*; this is implemented in most programming environments (in various different ways), but (subject to provisos about the minimum and maximum allowable integers in different implementations) the general properties of integers, and the results of operations such as addition, multiplication and equality, can be relied upon in all cases.

We have 'explained' the list operations above informally in plain English, but a formal definition would be needed in order to provide mathematical precision. One way of doing this is to specify a set of 'laws' that the list operations obey: for example, if we append an element onto the front of a list, and then take the head of the resulting list, we obtain the element we first started with. A complete abstract definition of a *List* ADT might be the following:

signature

Create	:		→ List
IsEmpty	:	List	→ Boolean
Head	:	List	→ Element
Tail	:	List	→ List
Append	:	Element × List	→ List

laws

IsEmpty(Create())
Head(Append(e,l)) = e
Tail(Append(e,l)) = l
NOT IsEmpty(l) ⇒ Append(Head(l),Tail(l))=l

The 'domain' of an ADT can be understood conceptually as the set of all possible objects of the type being defined. In computer terms it is the set of all possible values (or states) of variables of the given type. By generating a 'domain model' for an ADT, we can model it conceptually in terms of other ADTs. The process can be applied repeatedly until we arrive at basic abstract concepts that are considered to be universally understood and can be regarded as 'given'. In computer terms, given suitable software tools, domain modelling allows us to implement a high-level ADT using progressively lower-level ones, down to a level where we can use ADTs already provided, either by another researcher or 'off-the-shelf' by a software or hardware manufacturer (e.g. *Integer* and *Real* numbers). Software packages are increasingly providing explicit support for modelling using ADTs: these include modular and object-oriented programming environments, and more recently relational database systems such as INGRES (see Ryan, this volume).

Our definition of a *List* ADT encapsulates what is important about a generic list, while leaving out irrelevant details (such as the nature of the elements in any particular case). We can use it to construct lists of any type of basic object, or more complex entities such as lists of lists. In a similar way, we can construct ever more complex and specialised ADTs at progressively higher levels. Each ADT represents a model of a real-world object in a structural hierarchy.

As a simple example, consider a high-level ADT *SmallFirm* that provides an abstraction of a real-world small firm for a computer payroll system. For this purpose, all that is important about a firm is the employees within it: details such as the colour of the building in which the firm is housed are irrelevant. Thus *SmallFirm* might be modelled as an indexed set of employees, that is a set of employees each of whom is assigned a unique identifier such as an employee number. (Employees' names might not suffice for this purpose, as they are not necessarily unique.) The high-level ADT *SmallFirm* has been modelled in terms of the lower-level ADT *Employee*. This can now be modelled in turn: the important things about an employee might be their name, date of birth, grade and monthly salary. Many other attributes of a real-world employee, such as their height, sex, or hair colour, are irrelevant and do not form part of the abstraction. This level in the model introduces several new ADTs, namely *Name*, *Date*, *Grade* and *MonthlySalary* each of

1. A number of standard computer science text books address the topics of ADTs, e.g. Cleaveland (1986) and Harrison (1989). The concept is also elaborated more fully for the non-specialist in Ryan's paper in this volume.

which must be modelled in turn. But we are now nearing the level of basic objects that we can assume as given. Name, for example, might simply be modelled as a character string and *MonthlySalary* as a real number. Date might be given, or might need to be further modelled as a combination of day (an integer), month (one of twelve enumerated values) and year (an integer). The resulting structure could be specified formally as follows:

SmallFirm	=	EmployeeId $\xrightarrow{m}$ Employee		
Employee	::	n	:	Name
		dob	:	Date
		g	:	Grade
		ms	:	MonthlySalary
Name	=	CharacterString		
Grade	=	{MANAGINGDIRECTOR, TEAMLEADER, TECHNICALSTAFF, SUPPORTSTAFF }		
MonthlySalary	=	$\mathbb{R}$		

The notation used is the Vienna Development Method Specification Language (VDM-SL) (Jones 1986; Jones & Shaw 1990; Andrews & Ince 1991), one of a number of formal specification languages (FSLs) that have been developed and standardised by computer scientists in recent years (Ruggles 1990). FSLs are mathematically-based notations for software specification whose syntax and semantics are precisely defined. The development of a specification is an exercise in conceptual modelling. The use of a formal notation constrains the modeller to be disciplined in their thoughts, for the resulting specification has a precise and unambiguous meaning. When completed, the specification may be used as the starting point for a computer implementation of the abstract object being modelled. Indeed, some formal specification languages are themselves part of an entire 'formal development method', VDM-SL itself being part of the Vienna Development Method (VDM).

Note that in developing the example above, we have been working 'top-down' from *SmallFirm* to *Employee* and eventually to basic ADTs such as *Real* (denoted by the mathematical symbol $\mathbb{R}$), *Integer* and *CharacterString*. This is a useful approach for the conceptual modeller or the designer of a particular computer application. The developer of generic 'off-the-shelf' software, such as a database system, will wish to work at lower levels, identifying basic ADTs of general use and implementing them, and perhaps developing tools to enable the conceptual modeller (application programmer) to develop their own, customised higher-level ADTs on top of them.

The question of different styles of modelling is a separate but related one. The concept of an ADT underlies the currently popular 'object-oriented' style of modelling, in which frameworks of abstract objects are specified and proceed to communicate through the exchange of messages. An 'object' within an object-oriented framework (such as *Account0012345*, an abstraction of the real-world object 'John Smith's bank account') represents a particular instance of a class of objects (*BankAccount*) which essentially corresponds to an ADT. If the domain model of the ADT *BankAccount* includes, say, *CurrentBalance*, then the state of the object *Account0012345* will include the current balance of that account. An object-oriented system is essentially dynamic, so that John Smith's current balance may be updated at any time when a suitable message is received from another object within the system.

14.2 Spatial Objects, GIS, and Archaeology

Geographic Information Systems (GIS) are defined in the Core Curriculum of the American 'National Centers of Geographic Information and Analysis' (NCGIA) as Information Systems supporting geographical data, i.e. systems including procedures designed to support the capture, management, manipulation, analysis, modelling and display of spatially-referenced data. What is generally taken to distinguish them from other computer systems within which spatial data can be displayed and processed (e.g. CAD and statistics packages) is the ability to perform complex spatial operations and to generate new geographical data from existing data. A powerful function commonly incorporated in GIS systems is the ability to perform set-theoretic manipulations on sets of points, lines and areas so as to combine different types of spatial data and to visualise the result in terms of map 'overlays'.

A GIS can also be regarded as a computer system which supports basic ADTs with spatial attributes. In the same way that a computer language such as Pascal or a relational database system such as INGRES supports certain basic (abstract) data types such as *Character* and *Integer*, so a GIS supports basic spatial data types such as *Point*, *Line* and *Area*. A definition of the *Area* ADT might include the following operations:

UnionOf:	Area × Area → Area
IntersectionOf:	Area × Area → Area
Around:	Area → Area
NearTo:	Area → Area
Between:	Area × Area → Area
DirectionFrom:	Area × Direction → Area

This view of GIS means that they can be regarded not just as systems enabling spatially-related data to be manipulated and visualised, but also as support tools for conceptual modelling using objects with spatial properties. While it is the former that accounts for the sudden realisation of the great potential of GIS within archaeology (e.g. Allen *et al.* 1990), the latter is equally important (Ruggles forthcoming).

There is, however, a fundamental problem. This is the fact that no standard data structure model currently underlies GIS in general, in other words there is no set of universally agreed basic spatial ADTs. On one level this means that there is no guarantee that high-level data structured according to one model, and held within a particular GIS, will easily port to another GIS, or to a GIS of the future. On another level it means that the conceptual modeller cannot be assured that the basic 'bricks' underlying his or her model will not subsequently be replaced by different, incompatible ones as low-level spatial theory advances. The problem has been recognised for many years within the GIS community: for example, a group of experts concluded in 1983 that the absence of a coherent theory of spatial relationships 'hinders the use of automated GIS at nearly every point', (Boyle 1983). Two major outstanding problem areas are the following:[2]

- *The relationship between geometrical and topological properties*. An extreme approach to the modelling of spatial objects is the 'purely geometrical', in every spatial object is modelled in terms of the three basic ADTs *Point*, *Line* and *Area*. This sort of approach was quite common amongst early GIS implementations, and appears to lend itself easily to hierarchical modelling. However, it has the severe practical limitation that topological properties, such as whether lines intersect or points lie on lines, are not necessarily preserved under transformations, particularly changes of scale. Consider, for example, the problem of modelling two closed polygons with an edge in common. However carefully, and in whatever order, these are digitised, false gaps or slivers may occur when the data are displayed at a significantly larger scale. In working systems these problems are minimised by a number of more or less sophisticated fiddles.

  At the opposite extreme are 'totally topological' approaches in which each point, line or area in a map must be topologically related to all the others. An example of this approach is the 'cell complexes' of Frank and Kuhn (1986), in which there is effectively a single basic ADT, *Triangle*. The spatial attributes of any abstract object, however complex, are described in terms of a set of triangles tied together through various topological constraints. The problem here is that there is no possibility of hierarchical modelling, since the topological constraints must always be expressed at the lowest level. Every new geometrical object added to the system has to be topologically tied in at the outset, once and for all. A fruitful area of research is the investigation of 'halfway house' solutions that allow some geometrical objects to exist without necessarily being tied topologically to all other such objects, and permit hierarchical structuring.

- '*Fuzzy Operators*'. In general, the meaning of certain spatial operators, such as *Around* and *NearTo*, will depend on a particular set of circumstances, and may not always be precisely definable. It is a problem of the human-computer interface to attempt to tie this meaning down as precisely as is useful or necessary. A considerable amount of research has being directed to formalising spatial relations (e.g. Robinson *et al.* 1986; Haller, 1989; Angell *et al.* 1990).

The problem of fuzziness is not innately insurmountable, for specifying the result of a fuzzy operation 'as precisely as is useful or necessary' may itself be done precisely for any particular set of circumstances, and hence the result of the operation can still be formally specified (e.g. Newman *et al.* forthcoming). The problem is to agree on that specification, and hence to agree on the relevant low-level ADTs. The problem of geometry *vs* topology also concerns how best to specify the low-level ADTs upon which high-level spatial objects should be modelled, so as to achieve the greatest conceptual power and efficiency of implementation.

It is clear, then, that no universally agreed basic set of low-level spatial ADTs currently exists, nor is it definitely yet in sight. What, then, is the point in attempting to perform high-level conceptual modelling, or to build a complex spatial model within a GIS, when it appears to be founded upon quicksand? In a recent paper (Ruggles forthcoming) I have argued that there are considerable benefits in the simultaneous development of high-level abstractions within an application area such as archaeology and the low-level abstractions that underlie them within a field such as GIS. Furthermore, these benefits are mutual. Since archaeologists are extensively concerned with spatially referenced data and their interrelationships, they can make widespread use of low-level abstract spatial data structures within GIS. By constructing high-level abstract objects and being keen to implement them on a GIS in order to manipulate them and visualise their properties, archaeologists formulate requirements for the low-level abstractions that must underlie them, and hence help to drive research in GIS data structures itself. This will then result in improvements in 'off-the-shelf' software that are of direct benefit to archaeologists.

In the context of that argument I discussed the example of a simple terrain model. This example will be used again here, but reformulated and related strongly to the ADT concept.

14.3 Terrain models and domain models: a comparison of two approaches

Consider a simple terrain model that might be used by a landscape archaeologist to model locations within a geographical area under study. The entire model may be encapsulated in a single ADT *Terrain* which is an abstract model of the terrain of the area under study. The conceptual problem is to provide a domain model for this ADT in terms of progressively lower-level ADTs. The mathematical model so formed might then form the basis of direct (statistical) analysis. It might also form a specification that can be implemented within a GIS.

Every step in the modelling process reflects a particular type of approach that may be heavily influenced by existing paradigms: the possible types of approach and the resulting models may be very different indeed.

2. A further major problem area in GIS, and in spatial analysis generally, is that of scale change and generalisation. This concerns the fact that the representation of a geographical object may depend upon the scale of representation. On a large-scale map, for example, each house within a town might be represented by an area, whereas on a small-scale map the entire town might simply be represented as a point. A great deal of work has been undertaken in the area (e.g. Brassel & Weibel 1988; McMaster 1989). However, the problem is one of graphical representation for optimal visualisation. Data structures for generalisation are a refinement, or 'concretisation', of abstract spatial objects. If we accept the principle that the first priority is to understand and specify spatial objects and their interrelationships in the abstract, and only then to worry how best to represent them, then the generalisation problem is not our first priority. For this reason it is discussed no further here.

Nowhere is this clearer than at the very outset in the domain modelling process for the *Terrain* ADT. The approach of a cartographer, used to representing and visualising the terrain of an area as a contour map, might be to model the terrain as a set of contours, indexed by a unique identifier relating to elevation. Note that specific details such as the exact nature of a contour and the actual elevations for which contours will be included in the model are irrelevant at this stage: it is the general form of the model that is important. Mathematically, we would represent this as a partial function, or 'mapping', from the set of (all possible) contour identifiers to the set of (all possible) contours, in other words a function that associates at most one contour with each possible contour identifier. In VDM this is expressed as follows:

$$\text{Terrain} = \text{ContourId} \xrightarrow{m} \text{Contour}$$

An archaeologist, on the other hand, might be heavily influenced by the desire to model terrain characteristics at particular locations in order to correlate them with human activity patterns. This could lead to a view of the entire terrain model as a set of 'point terrain models', and hence to a first refinement of the domain model of the *Terrain* ADT as an indexed set of point terrain models. Following this approach we would model the terrain of an area mathematically as a mapping (partial function) from the set of points within the area to the set of point terrain models. In VDM:

$$\text{Terrain} = \text{PointId} \xrightarrow{m} \text{PointTerrain}$$

As in the case of the cartographer's model, more specific questions, such as the exact set of points at which point models will be required, are irrelevant at this level of abstraction, leaving the theoretician to concentrate on broad issues such as whether the general manner in which the terrain of an area has been modelled is a useful one. Note also that the domain of the mapping is actually a set of identifiers of points rather than a set of points themselves. On grounds of elegance and efficiency it is desirable to restrict the domains of mappings to be sets of tokens (such as identifiers) rather than more complex objects. This is analogous to arranging entity types in third normal form. A separate data structure would map point identifiers onto their co-ordinates for cross-reference.

Both the approaches described above have produced a new, lower-level ADT. The cartographer has introduced the ADT *Contour* and now has to model its structure. It has been specified that only a single contour may have a given identifier, and it is intended that an identifier should correspond to a given elevation. Thus a 'contour' is in fact an abstraction of all locations in the area whose elevation is the elevation in question. These break down into discrete sets of locations joined by continuous lines, some forming closed loops and some running to the edge of the geographical area in question. Thus we might specify

$$\text{Contour} = \text{Line-set}$$

where *Line*, representing an open or closed continuous line, is regarded as a basic spatial ADT.[3]

In the alternative approach we now need to provide a domain model for the ADT *PointTerrain*. A degenerate model, yet one that is entirely valid, is simply the elevation of the point in question, in other words

$$\text{PointTerrain} = \text{Elevation}$$

This, of course, produces the basis for a standard digital terrain model. The archaeologist's model, however, may be considerably richer. To model the terrain at a given location, we might use four mutually independent indexes: elevation, slope, aspect, and 'terrain form index', an indicator suggested by Kvamme (1989). Such an approach is well exemplified in Kvamme and Jochim's analysis of the distribution of Mesolithic sites in southern Germany (1988). The elevation may simply be modelled as a real number (in some unit of measurement), the slope as an angle between 0° (horizontal) and 90° (vertical), the aspect as an azimuth (i.e. an angle between 0° and 360°) and the terrain form index as a non-negative number. The point terrain is a combination of these four indexes. In VDM:

$$\begin{array}{lcll}
\text{PointTerrain} & :: & \text{e} & : \text{Elevation} \\
 & & \text{s} & : \text{Slope} \\
 & & \text{a} & : \text{Aspect} \\
 & & \text{tfi} & : \text{TerrainFormIndex}
\end{array}$$

$$\begin{array}{ll}
\text{Elevation} & = \mathbb{R} \\
\text{Slope} & = \text{Angle} \\
\quad \textbf{inv } s \triangleq s \geq 0 \wedge s \leq 90 & \\
\text{Aspect} & = \text{Angle} \\
\text{TerrainFormIndex} & = \mathbb{R}^{+} \\
\text{Angle} & = \mathbb{R} \\
\quad \textbf{inv } a \triangleq a \geq 0 \wedge a < 360 &
\end{array}$$

The basic spatial ADT upon which the objects modelled in this example are based is *Point*, but in other cases use might also be made of *Line* (e.g. when modelling communication routes) or *Area* (e.g. when modelling surface geology).

14.4 The importance of hierarchical structuring

The analytical functionality and visualisation potential supported within the current generation of GIS is largely based on flatly-structured contour or digital terrain data. This limitation is at the heart of many of the observed error and inconsistency problems, both in analysis (different ways of implementing analytical procedures in different GIS systems can significantly affect the results of the analyses — e.g. Kvamme 1990) and visualisation (e.g. spurious 'gaps and slivers' may appear when data are displayed at different scales, as mentioned above).

Hierarchical structuring, while not completely eliminating such problems, can bring them under control by making them directly manageable. The key is that specification of an ADT takes place at one level, and its implementation at a lower level. An implementer is faced with the task of providing a working ADT that conforms precisely to the functional

3. Because of possible confusion in the nomenclature it might be felt that a name such as *ContourLineSet* is more appropriate than the name *Contour* to describe the ADT in question. Such considerations are important because of the need to communicate an appreciation of the nature of a model to others who have experience of the real-world objects being abstracted. However, they need not concern us further here.

specification provided. The problem of how to determine, for example, the terrain form index at a given point, is separated into one of specifying the ADT *TerrainFormIndex*, a problem in the domain of the archaeologist, and one of implementing it. The specifier specifies the desired properties of a number that reflects the roughness of the terrain at a given point: the implementer then decides whether this can be satisfied, for example, by taking the range of elevations within a given radius of the point, or the standard deviation of sample elevations within a given radius, or by some other means. A formal specification will include analytical operations that can be performed upon terrain form indices, including their relationship to (and hence their determination from) other terrain data, which an implementer must satisfy within given tolerances. The problem of errors and inconsistencies between different implementations, should thus be avoided, or at least confined to within the specified tolerances. The archaeologist can also, of course, include the index as an integral part of a higher-level conceptual model and discuss its abstract properties separately from consideration of implementation difficulties.

Hierarchical structuring also allows high-level abstract modelling to take place as a theoretical exercise, without the constraint of any necessity for immediate implementation in a computer information system. (By recognising abstractions such as *Terrain* itself as high-level ADTs we have already taken an important step along this road.) Indeed, it is important that implementation constraints be cleanly separated from theoretical issues; otherwise there will be a tendency to limit theoretical concepts to relatively simple, readily implementable ones. A good example from the days before GIS is the modelling of areas of social influence using Thiessen polygons. Within a GIS far more complex areas may be analysed and visualised. While the introduction of GIS has made considerably more complex spatial objects relatively easy to handle, it is imposing an artificial (and probably temporary) constraint to limit one's consideration to conceptual objects that are readily implementable even in the current generation of GIS.

A further example given by Ruggles (forthcoming) concerns the problem of obtaining a measure of the view from a point location, a factor that may be of great importance in analysing distributions of archaeological phenomena with respect to the landscape. A simple criterion often used in the past has been view catchment, defined as the percentage of the terrain within a given distance of the point in question that is visible from it. Before the days of GIS this simple scalar measure was determinable, albeit with great tedium, from physical maps. However the chosen distance is of course arbitrary, so a more representative measure might be

$$\text{ViewCatchment} \quad = \quad \text{Distance} \xrightarrow{m} \text{Percentage}$$

where the domain of the mapping is a set of discrete values at regular intervals. In the limit as the intervals tend to zero, view catchment tends to a continuous function on $\mathbb{R}^+$.

Then again, studies in archaeoastronomy and elsewhere have emphasized the importance in certain contexts of the orientation of visibility, that is not just the total view from a location but how the horizon distance varies with azimuth. This implies a still more complex general abstract conception of view catchment. Both Fraser (1983) and Ruggles (1984) used a small number of discrete distance categories in their analyses, owing to practical limitations, but in the abstract *ViewCatchment* might now be defined as follows:

$$\text{ViewCatchment} \quad = \quad \text{Azimuth} \xrightarrow{m} \text{HorizonDistance}$$

Note that the previous measures of view catchment can be deduced from this more general one. Further factors, such as the visibility of landmarks (e.g. mountain peaks), might be included in an even more complex abstract conception of view catchment.

14.5 Conclusions

One of the most basic principles that should consistently underlie conceptual modelling is that of hierarchical abstraction. The concept of an Abstract Data Type lies at the heart of this approach. At the highest level in a model are ADTs representing abstractions, possibly highly complex in structure, of objects in the real world that are of special interest in a particular field of enquiry. At the lowest level are ADTs representing abstractions of basic, fundamental objects that are both widely understood and widely applicable. At progressively higher levels in the hierarchy, objects become progressively narrower in their field of interest. Below certain levels it makes sense to attempt to standardise ADTs so that 'off-the-shelf' concepts can be used in a wide variety of conceptual models and 'off-the-shelf' ADTs are widely available in commercial software, thus facilitating the portability and future-proofing of computer implementations of models between different software systems.

The interaction between the development of conceptual models in an application area, which involves the specification of high-level ADTs, and the development of commercial system software of wide applicability, which involves the specification and implementation of low-level ones, is two-way. Not only can and should techniques of data modelling originating in computer science and mathematics be incorporated as powerful tools in high-level model building; it is also the case that the attempted description (implementation) of high-level ADTs in terms of low-level ones will considerably aid research attempting to clarify and standardise concepts at the low level.

Nowhere is this more true than in GIS applications in archaeology. GIS are concerned with low-level ADTs with spatial attributes. Archaeology is centrally concerned with objects in the physical and human environment and their interrelationships, and in particular with the spatial attributes of these objects. Thus archaeology provides arguably one the richest possible application areas for developing high-level spatially referenced ADTs, and hence for driving requirements for standardised low-level ones.

An important new development is that of modelling low-level ADTs with temporal attributes. The area of 'temporal databases' and 'temporal GIS' is beginning to attract a good deal of attention (Armstrong 1988; Langran 1989; Worboys 1990). Ultimately, one can envisage 'spatio-temporal information systems', which provide a basic framework of low-level ADTs with temporal as well as spatial attributes. Few application areas are in a better position to drive developments in this area than archaeology (see Castleford, this volume).

In the meantime, one of the most immediate needs is for GIS, like programming environments and database management systems, to begin to support user-defined ADTs, thus enabling the application programmer to construct a structural hierarchy and to implement complex abstractions. And one of the most immediate needs for archaeologists is to clarify and formally specify their high-level concepts in the form of ADTs, particularly in areas such as landscape studies and in temporal modelling. Here, high-level abstractions in specific applications areas have a crucial role to play in influencing the development of a standard structure of fundamental spatio-temporal concepts to support them, both on and off the computer.

Acknowledgements

I am most grateful for the helpful comments of the referee.

References

Allen, K.M.S., S.W. Green & E.B.W. Zubrow (eds.) 1990. *Interpreting Space: GIS and Archaeology*, London, Taylor & Francis.

Andrews, D. & D. Ince 1991. *Practical Formal Methods with VDM*, London, McGraw-Hill.

Angell, R., D.J. Medyckyj-Scott & I. Newman 1990. *The Development of a User-Interface for the Handling of Spatial Language Queries in a Geographic Information Retrieval System, 1: Theoretical and Conceptual Issues*, Research Report No. 12, University of Leicester and Loughborough University, Midlands Regional Research Laboratory.

Armstrong, M.P. 1988. "Temporality in spatial databases", *Proceedings of GIS/LIS 1988*, 2: 880–889.

Boyle, A.R. 1983. *Final report of a conference on the review and synthesis of problems and directions for large-scale Geographic Information System Development*, ESRI, Redlands CA, NASA Contract NAS2-11346.

Brassel, K. & R. Weibel 1988. "A review and framework of automated map generalisation", *International Journal of Geographical Information Systems*, 2(3): 229–244.

Cleaveland, J.C. 1986. *An Introduction to Data Types*, London, Addison-Wesley.

Frank, A.U. & W. Kuhn 1986. "Cell graphs: a provable correct method for the storage of geometry", *Proceedings of the 2nd. International Symposium on Spatial Data Handling*, Seattle, Washington: 411–436.

Fraser, D. 1983. *Land and Society in Neolithic Orkney*, British Archaeological Reports (British Series) 117, Oxford, British Archaeological Reports.

Haller, S. 1989. "Spatial relations representation and locative phase generation in a map context", *National Centre for Geographic Information and Analysis*, State University of New York at Buffalo, Technical Paper 89–14.

Harrison, R. 1989. *Abstract Data Types in Modula-2*, London, John Wiley.

Jones, C.B. 1986. *Systematic Software Development using VDM*, London, Prentice-Hall.

Jones, C.B. & R.C. Shaw 1990. *Case studies in Systematic Software Development*, London, Prentice-Hall.

Kvamme, K.L. 1989. "Geographic information systems in regional archaeological research and data management", in M.B. Schiffer (ed.), *Archaeological Method and Theory*, 1, Tucson, University of Arizona Press: 139–203.

Kvamme, K.L. 1990. "GIS algorithms and their effects on regional archaeological analysis", in K.M.S. Allen, S.W. Green & E.B.W. Zubrow (eds.): 112–125.

Kvamme, K.L. & M.A. Jochim 1988. "The environmental basis of Mesolithic settlement", in C. Bonsall (ed.), *The Mesolithic in Europe*, papers presented at the 3rd International Symposium, Edinburgh 1985, Edinburgh: 1–12.

Langran, G. 1989. "A review of temporal database research and its use in GIS applications", *International Journal of GIS*, 3(3): 215–232.

McMaster, R. (ed.) 1989. "Numerical generalization in cartography", *Cartographica*, 26(1), Monograph 40.

Newman, I.A., D. Medyckyj-Scott, C.L.N. Ruggles & D.R.F. Walker forthcoming. "Handling the spatial component of a metadata query — a formal approach", submitted to *International Journal of GIS*.

Robinson, V.B., M. Blaze & D. Thongs 1986. "Representation and acquisition of a natural language relations for spatial information retrieval", *Proceedings of the 2nd International Symposium on Spatial Data Handling*, Seattle, Washington: 472–87.

Ruggles, C.L.N. 1984. "A new study of the Aberdeenshire Recumbent Stone Circles, 1: Site data", *Archaeoastronomy* (supplement to Journal for the History of Astronomy), 6: S55–79.

Ruggles, C.L.N. (ed.) 1990. *Formal Methods in Standards*, Berlin and London, Springer-Verlag.

Ruggles, C.L.N. forthcoming. "A fair exchange: the mutual benefits of theory development within Archaeology and within GIS", paper delivered at TAG 90, Lampeter, December 1990, and submitted to *Journal of Archaeological Science*.

Worboys, M.F. 1990. *Reasoning about GIS using temporal and dynamic logics*, Research Report No. 18, University of Leicester and Loughborough University, Midlands Regional Research Laboratory.