--

SPATIAL REASONING USING THE QUAD TESSERAL REPRESENTATION

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

Department of Computer Science,
The University of Liverpool,
Chadwick Building,
P.O. Box 147,
Liverpool L69 3BX,
England.
email: f.p.coenen@csc.liv.ac.uk
Tel: 0151 794 3698
Fax: 0151 794 3715

ABSTRACT

A review of the application of the quad tesseral representation to support spatial reasoning is presented. The principal feature of the representation is that it linearises multi-dimensional space, while still providing for the description of individual objects within that space and the relationships that may exist between those objects (in any direction and through any number of dimensions). In addition the representation is supported by an arithmetic which allows the manipulation (translation etc.) of spatial objects. Consequently, when incorporated into a spatial reasoning system, all necessary processing can be implemented as if in only one dimension. This offers two significant advantages over more conventional multi-directional approaches to spatial reasoning. Firstly, many of the concerns associated with the exponential increase in the number or relations that need to be considered (as the number of dimensions under consideration increases) are no longer relevant. Secondly, the computational cost of manipulating and comparing spatial objects remains static at its one dimensional level, regardless of the number of dimensions under consideration.

KEYWORDS: Spatial Reasoning, Spatial Linguistics, Quad Tesseral Addressing.

October 9, 1997

--

SPATIAL REASONING USING THE QUAD TESSERAL REPRESENTATION

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

Department of Computer Science,
The University of Liverpool,
Chadwick Building,
P.O. Box 147,
Liverpool L69 3BX,
England.
email: f.p.coenen@csc.liv.ac.uk
Tel: 0151 794 3698
Fax: 0151 794 3715

1. INTRODUCTION

There is a significant body of research work concerned with the representation of two-dimensional space to support spatial reasoning using Constraint Satisfaction Problem (CSP) techniques. Examples include the work of Freksa (1991), Hernández (1991) and Retz-Schmidt (1988). Much of this work is based on a two-directional (X-Y) view of the space under consideration and concentrates on the adaptation of well tried and tested one-dimensional (temporal) techniques (such as those proposed by Allen (1991) or Dechter et al. (1991) to cite two examples) to address two-dimensional reasoning. However, the scaling up of such one-dimensional techniques to encompass two-dimensional reasoning results in:

1) A combinatorial explosion of the number of relations that must be considered and,
2) A corresponding increase in the computational cost of manipulating spatial objects (usually implemented using trigonometric or matrix techniques).

These disadvantages are compounded when the scaling is extended to encompass three and four dimensional reasoning.

In this paper we show how an alternative representation can be used to support spatial reasoning. This representation is based on the concept of quad tesseral addressing (Diaz and Bell 1986, Gargantini 1983) which, it is suggested, serves to avoid many of the concerns associated with traditional approaches to spatial reasoning. The features of this representation can be summarised as follows:

- The linearisation of space.
- Data compression through the grouping of sequences of addresses in terms of a start and end address.
- As a result of the above, computationally inexpensive implementation of "set operations" and their comparison.
- A supporting arithmetic which facilitates translation through a tessellated space in any direction (and through any number of dimensions).
- Applicability to N-dimensional space (although in this paper we will concentrate only on two-dimensional space).

A more comprehensive discussion of the features of the quad tesseral representation is presented in Section 2. When incorporated into a spatial reasoning system the representation offers the following advantages over more conventional multi-directional approaches:

- Arbitrarily shaped objects, regardless of their complexity, can be defined simply and concisely in terms of sets of addresses (regardless of the number of dimensions under consideration).

- All relationships between spatial objects can be expressed as if in one dimension and hence many of the concerns associated with the number of relations that otherwise require consideration are removed.
- The supporting arithmetic facilitates the effective description of the relationships that may exist between spatial objects in terms of tesseral *offsets*.
- Computationally efficient comparison of objects can be achieved using the supporting arithmetic and knowledge of the linearisation/sequencing.

These advantages will be expanded upon in Sections 3, 4 and 5. The description of spatial objects will be considered in Section 3, and the definition of the relationships that may be required to exist between objects in Section 4. In Section 5 consideration will be given to the manipulation of spatial objects where the required relationships are interpreted as constraints.

The proposed technique has been incorporated into a spatial reasoning system which in turn has been successfully applied to a number of application areas including: Geographic Information System (GIS), Environmental Impact Assessment (EIA) and timetabling. Details of these applications are presented in Section 6.

2. THE QUAD TESSERAL REPRESENTATION

Tesseral addressing is based on ideas concerning hierarchical clustering developed in the 1960s and 1970s to improve data access times (Morton 1966, see also Samet 1984), and atomic isohedral (same shape) tiling strategies developed in the 1970s and 1980s concerned with group theory (Grunbaum & Shephard 1987). These two strands were brought together in the early 1980s when a *tesseral arithmetic* was defined for the representation (Holroyd 1983). There are many excellent works that deal in detail with the concept of tesseral addressing (see Diaz and Bell 1986, Gargantini 1983). However, so that a complete understanding of the spatial reasoning technique described can be gained, it is essential that the basic ideas behind the quad tesseral representation are outlined here.

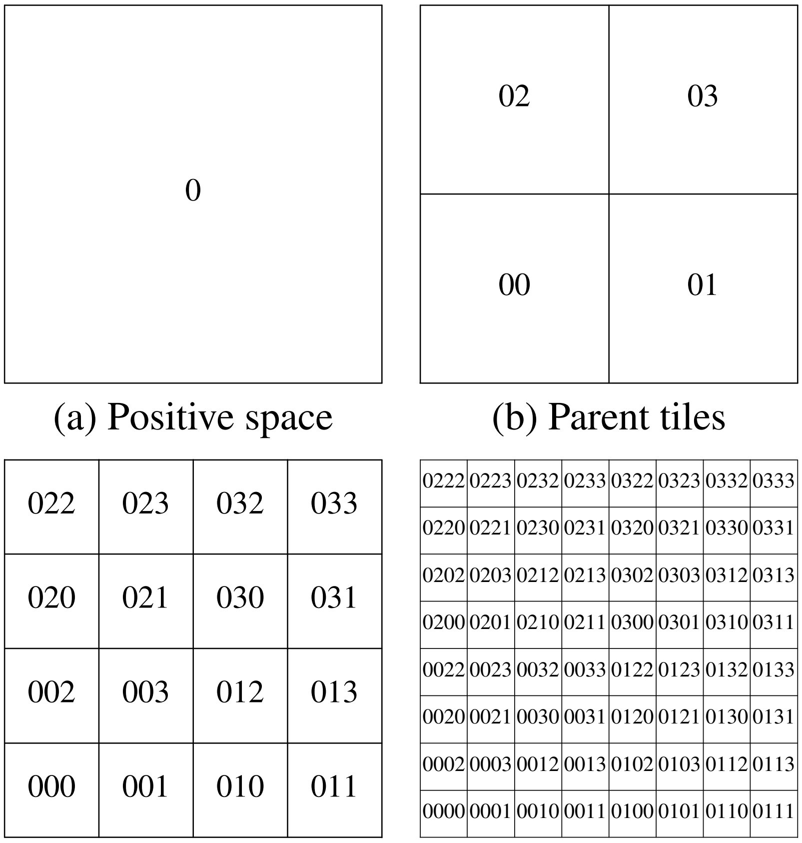
2.1. Tesseral Addressing

At its simplest tesseral addressing can be viewed as a method of labelling `tiles` in a two-dimensional space. The addressing mechanism is as follows. Given a square area (Figure 1(a)) we give this the address `0`. If we now quarter this area (i.e. *tessellate* it) each quadrant (tile) will be labelled as shown in Figure 1(b) assuming that the origin is located at the bottom left hand corner. If we now also quarter these *parent* tiles we will produce *child* tiles as shown in Figure 1(c). Note that each child tile takes an address made up of a left hand symbol corresponding to its parent's tesseral address and a right hand symbol corresponding to its position in relation to its siblings (i.e. `0`, `1`, `2` or `3`). In a similar manner we can produce grandchild nodes (Figure 1(d)) and so on until some desired depth of tessellation is arrived at with new tesseral addresses generated by always appending to the right of the parent tesseral address.

Using additional symbols (e.g. `4`, `5`, `6` and `7`) we can use a similar approach to address 3-dimensional space, and by adding further symbols (`8`, `9`, `A`, `B`, `C`, `D`, `E` and `F`) address 4-dimensional space.

2.2. Linearisation and the Morton sequence

One of the principal advantages of the quad tesseral representation, with respect to spatial reasoning, is that the addressing mechanism serves to linearise the object space of interest. If we consider the addresses presented in Figure 1(d) to represent integers to the base four we can convert these to decimals (Figure 2(a)) which then describe what is termed the Morton sequence (Figure 2(b)) after Morton (1966). In this paper sequences of tesseral addresses will be indicated using



(a) Positive space (b) Parent tiles

(c) Child tiles (d) Grand-child tiles

Figure 1: *Tesseral addressing*

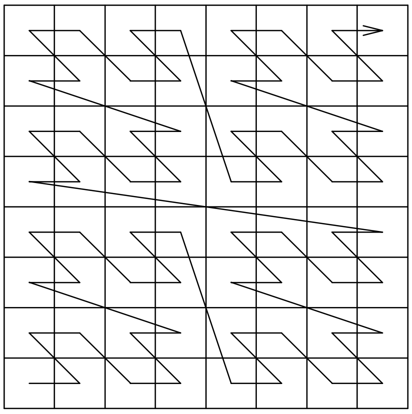
the infix operator "..", where the prefix operand represents the start address for the sequence and the postfix operand the end address. Thus the sequence given in Figure 2 would be described as `{0000..0333}`. It is important to appreciate that such sequences represent sets (or sub-sets) of addresses - hence the set notation.

2.3. Tesseral Addition

A further advantage offered by the quad tesseral representation is that it has an arithmetic (Holroyd 1983) which facilitates translation through the space in any desired direction simply by adding a tesseral displacement. From Figure 1(b) we can see that to move (translate) from tile `00` to the right we must add `1`; similarly to move up or diagonally to the top-right hand corner of the

42	43	46	47	58	59	62	63
40	41	44	45	56	57	60	61
34	35	38	39	50	51	54	55
32	33	36	37	48	49	52	53
10	11	14	15	26	27	30	31
8	9	12	13	24	25	28	29
2	3	6	7	18	19	22	23
0	1	4	5	16	17	20	21

(a) Morton numbering



(b) Morton sequence

Figure 2: *Morton sequencing*

object space we add `2` or `3` respectively. By generating an addition table (Table 1) we can determine the result of adding any tesseral displacement to any tesseral address.

+	0	1	2	3
0	0	1	2	3
1	1	10	3	12
2	2	3	20	21
3	3	12	21	30

Table 1: *Tesseral addition table*

The generation of this table is well understood and is described in Bell et al. (1983). Note that in the table where a look up gives two digits the left most digit represents a carry. Note also that multi-digit addition is performed using the place system in a fashion exactly analogous to ordinary (base 10) arithmetic. Thus, from Figure 1(d), if we wish to translate 3 tiles to the right and 2 tiles up from the origin we will arrive at tile `0031`. If we wish to make the same translation from (say) tile `0023` we must add the tesseral displacement `0031` to the address `0023`. From the table, this will produce the expected tesseral address `0302`.

Tesseral subtraction, although not described here, can be defined in a similar manner.

2.4. Negative Tesseral Addresses

Using tesseral addition we can translate to the right, upwards and diagonally in a broad "north-easterly" direction. To translate in any other direction it is necessary to add a negative tesseral address. Thus to support translation in all directions we also need to label the negative quadrants of Cartesian space. In addition this must be done in such a manner that the arithmetic is still supported. In Figure 1(a) we labelled the positive quadrant `0` which gave rise to the parent tesseral addresses `00`, `01`, `02` and `03` (Figure 1(b)). Assuming a constant depth of resolution (i.e. all tesseral addresses are made up of the same number of symbols), and with reference to Table 1, to move from tile `01` to tile `00` we must add the tesseral offset `11` (ignoring any carry beyond the address length). Thus, in Figure 1(b), the tile immediately to the left of tile `00` must be tile `11` and hence the quadrant immediately to the left of the positive quadrant must take the label `1`. Using the same process of deduction we can also determine the labels for the remaining two negative quadrants (Figure 3(a)). Thus the labelling of the first quadrant can now be extended to the entire Cartesian space (see Figures 3(b) and 3(c)). As a result it is possible to translate, in any direction, through a tessellated space by applying an appropriate tesseral displacement.

1	0
3	2

(a) Cartesian Space

12	13	02	03
10	11	00	01
32	33	22	23
30	31	20	21

(b) Parent tiles

122	123	132	133	022	023	032	033
120	121	130	131	020	021	030	031
102	103	112	113	002	003	012	013
100	101	110	111	000	001	010	011
322	323	332	333	222	223	232	233
320	321	330	331	220	221	230	231
302	303	312	313	202	203	212	213
300	301	310	311	200	201	210	211

(c) Child tiles

Figure 3: *Tesseral addressing in all four quadrants*

Again the negative labelling can be extended to three and four dimensions using appropriate symbols.

3. SPATIAL OBJECTS

Spatial reasoning is concerned with the interrelationships that exist between objects that have spatial characteristics. Such objects are considered to exist in some N-dimensional *object space* whose dimensions are sufficiently large so as to ensure that all relevant objects are contained somewhere within it. Further we can identify a number of attributes that such objects can take:

- Name
- Shape
- Location space.

The first acts as an identifier for the object and serves to group together the remaining attributes under a single unifying label. In the following text we will use the symbols `X` and `Y` to indicate such labels. The remaining attributes will be discussed in further detail in the following two subsections.

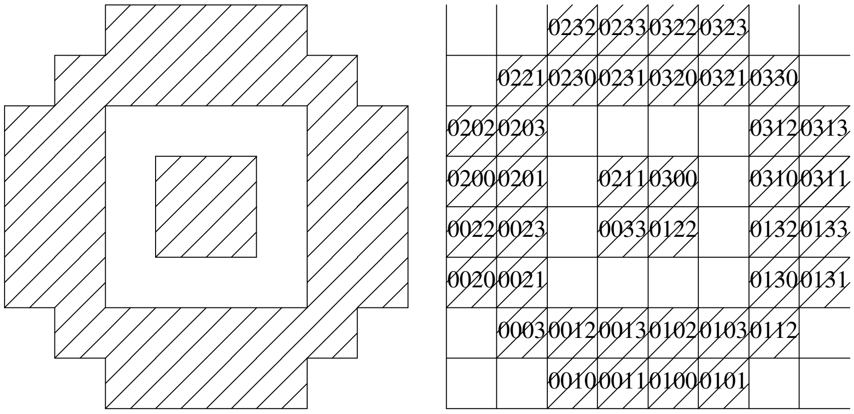
3.1. Shape Attributes

A shape attribute describes the shape of a spatial object. To this end we can identify two types of object, which we will refer to as *tile* and *zone* objects. A tile object, as its name suggests, occurs at a single tesseral address. In temporal reasoning systems such objects are usually referred to as *instantaneous* or *point* objects (Dechter et al. 1991, Bruce 1972). Note, however, that these addresses, although treated primitively, have as many dimensions as the object space: it is this feature that enables the straightforward linearisation of the space. Zones are then spatial objects which occur over more than one tesseral address. In temporal reasoning terms such objects are often referred to as *intervals* (after Allen 1983).

In conventional systems the shape of a zone object is usually described by defining its boundary in terms of a set of vectors which, in turn, can be defined in terms of their start and end Cartesian coordinates or in terms of a start point, direction and length. However, using this approach, it is computationally expensive to determine the relationships that exist between such objects. Using a tesseral representation any arbitrarily shaped object can be described and, more importantly, related to other similarly defined objects.

To obtain the tesseral description of a two-dimensional object, given some object space, we place the object so that the local origin (the south-west corner address) of the minimum bounding box surrounding the object coincides with the origin of the object space. The shape of the object can then be defined in terms of the set of tesseral addresses "included" in the object. In the case of a tile object this will, of course, always be the zero tesseral address. In the case of a zone object this will comprise a set of single and/or sequences of tesseral addresses. For example if we wish to describe the shaded "doughnut" shape given in Figure 4(a), the shape would be placed within our object space as indicated in Figure 4(b) and the appropriate tesseral addresses obtained, i.e. the set `{00003..000023, 00033..00103, 00112, 00122, 00130..00203, 00211, 00221, 00230..00300, 00310..00330}`. Note that, from the "size" (number of digits) of the addresses, we have assumed a 16×16 object space. Where the size of the object space is unknown we can select an address size, up to some machine dependent maximum, which will guarantee the inclusion of the required objects.

Thus from the above it is apparent that the tesseral representation allows the description of any spatial object (contiguous or otherwise) and regardless of the complexity of its design. In the following text we will use the symbol `XS` and `YS` to indicate the set of tesseral addresses describing the shape attribute associated with the objects `X` and `Y` respectively.



(a) **(b)**

Figure 4: Shape attribute definition.

3.2. Location Space Attributes

Every object has a location space attribute associated with it which describes the part of the object space in which the object may be said to exist. This is defined in a similar manner to that used to describe an object's shape attribute, i.e. using single and/or sequences of tesseral addresses. In the following text we will use the symbols `XL` and `YL` to indicate the location attributes associated with the objects `X` and `Y` respectively.

The relationship between the shape and location space attributes associated with an object defines the number of possible *candidate* locations for that object. This is described in terms of a list of sets of tesseral addresses. In the following text we will use the symbols `XC` and `YC` to indicate the candidate lists of sets of tesseral addresses for the objects `X` and `Y` respectively.

Given an object definition in the form of a predicate:

```
object(X,XS,XL)
```

we can generate the set `XC` as follows:

1. Initially XC is an empty list, {}.

2. For each $t \in XL$, $T = t+XS$

2. If $T \subseteq XL$ add T to XC.

where t is a single tesseral address belonging to a set of tesseral addresses, T is a set of tesseral addresses which may represent an appropriate location for the object X, and XC is the list of location sets of addresses associated with X. If the number of elements in the list XC (n(XC)) equals zero the object is not physically realisable. If n(XC) = 1 the object has only one possible location and hence the object is described as a *fixed* object. Otherwise, (n(XC) > 1) the object has a number of possible candidate locations and is referred to as a *free object*.

The above process can best be illustrated by considering the following examples which assume a 4×4 object space (see Figure 5.

EXAMPLE 1 (Fixed tile object):

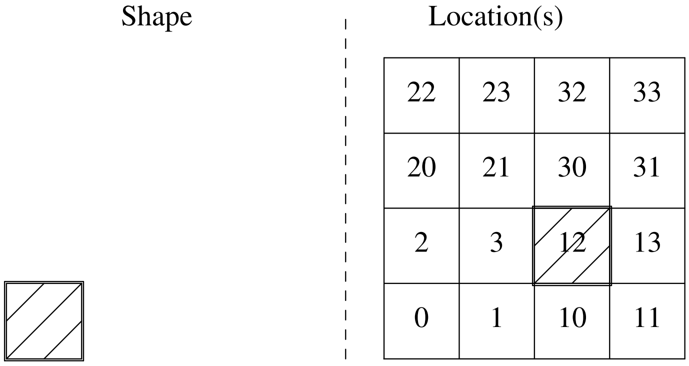
Consider a tile object X with a location space comprising a single address ({012}) (the shape attribute for a tile object is also a single address (XS = {000}). As a result of applying the shape attribute to the elements of XL only one possible location will be produced and hence X is a fixed tile object.

```
object(X, {000}, {012})
∀t∈XL+XS => 012+{000} = {012}
{012)⊆{012}
XC = {{012}}, n(XC) = 1
```

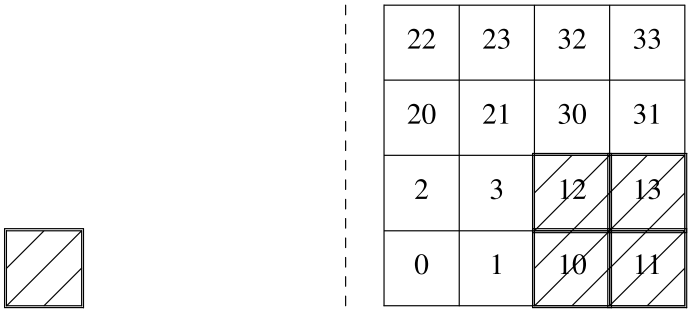
EXAMPLE 2 (Free tile object):

Given another tile object, and a location space comprising more that one element we can determine a number of possible candidate locations for the object - four in this case.

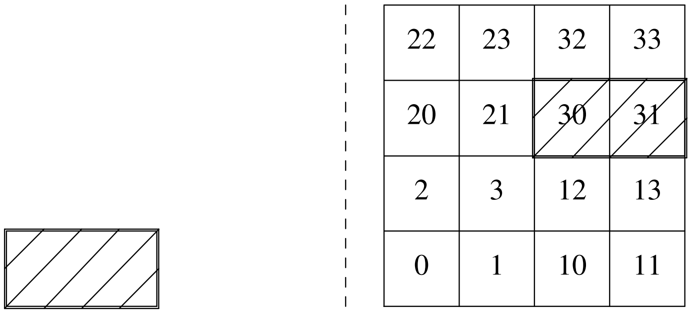
```
object(X, {000}, {010..013})

∀t∈XL+XS => 010+{000} = {010}
            011+{000} = {011}
            012+{000} = {012}
            013+{000} = {013}

{010}⊆{010..013}
{011}⊆{010..013}
{012}⊆{010..013}
{013}⊆{010..013}

XC = {{010}, {011}, {012}, {013}}, n(XC) = 4
```



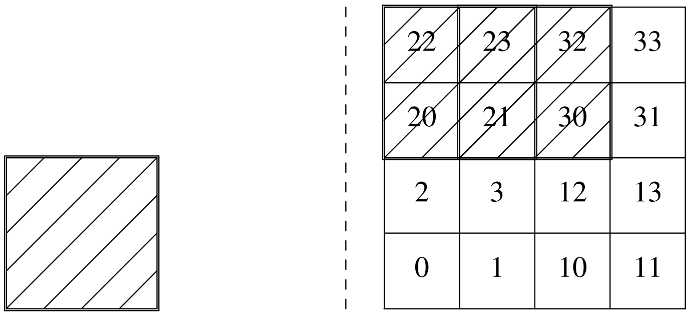
(a) EXAMPLE 1 (Fixed tile object):



(b) EXAMPLE 2 (Free tile object):



(c) EXAMPLE 3 (Fixed zone object):



(d) EXAMPLE 4 (Free zone object):

Figure 5: *Relationship between shape and location attributes*

EXAMPLE 3 (Fixed zone object):

Similarly given a zone object (i.e. an object whose shape attribute is comprised of more than one address) and a location space we again apply the shape attribute to the elements of `XL` to produce a number of candidate locations. Note that in this case when we add `XS` to `{031}` one of the addresses produced, `0120`, is outside the object space and thus cannot form part of a candidate location. As a consequence only one "candidate" location is found and the object is described as a fixed object.

```
object(X, {000..001}, {030..031})

∀t∈XL+XS => 030+{000..001} = {030..031}
            031+{000..001} = {031, 0120}

{030..031}⊆{030..031}
{031, 0120}⊆{030..031}

XC = {{030..031}}, n(XC) = 1
```

EXAMPLE 4 (Free zone object):

Alternatively, if a number of candidate locations result from the calculation, the object is considered to be a free object. Note also that in this example the tesseral addresses `0200`, `0201`, `0210` and `0211` are all outside the object space and hence these addresses can not form part of any candidate location, leaving us with two candidate locations.

```
object(X, {000..003}, {020..030, 032})

∀t∈XL+XS => 020+{000..003} = {020..023}
                  021+{000..003} = {021, 023, 030, 032}
                  022+{000..003} = {022..023, 0200..0201}
                  023+{000..003} = {023, 032, 0201, 0210}
                  030+{000..003} = {030..033}
                  032+{000..003} = {032..033, 0210..0211}
```

```
{020..023}⊆{020..030, 032}
{021, 023, 030, 032}⊆{020..030, 032}
{022..023, 0200..0201}⊆{020..030, 032}
{023, 032, 0201, 0210}⊆{020..030, 032}
{030..033}⊆{020..030, 032}
{032..033, 0210..0211}⊆{020..030, 032}

XC = {{020..023}, {021, 023, 030, 032}}. n(XC) = 2
```

4. SPATIAL RELATIONSHIPS

There is a rich body of research work concerned with the "linguistics" of spatial reasoning - the taxonomy whereby the relationships between spatially distributed entities can be classified. This has its roots in the 26 one-dimensional relations that can exists between points and/or intervals, namely 13 interval-interval relations, 5 interval-point relations, 5 point-interval relations and 3 point-point relations (see Allen (1983) and Hamblin (1971) for further discussion). For many applications we also need to consider the negation of such relations. These can be expressed simply by disjoining groups of relations. However, for practical purposes disjunctions are awkward and cumbersome to manipulate and thus it is more computationally expedient to include a further 26 negative relations. Some authors (for example Freksa 1992) also identify sets of partial relations which apply given incomplete data. Further, to make relationships more expressive the concept of offsets (Coenen et al. 1994) can be included so that an object X can be said to be a certain distance before or after an object Y.

When scaling up this taxonomy to address the relationships that can exist in two-dimensional space it is not effective to extend the above 26 one-dimensional relationships to two directions as this will give rise to 228 relations (13×13 interval-interval relations, 5×5 interval-point relations, 5×5 point-interval relations and 3×3 point-point relations). A significant amount of work has been undertaken to reduce the number of relations that need to be considered when reasoning about multi-dimensional space. One approach is to define all relationships in terms of point-point relationships. This is an approach made popular by many point-based temporal reasoning systems (see Bruce (1972), Ladkin (1987, 1992), Dechter et al. (1991) and Maiocchi (1992)). Other authors (Chang et al. 1987, Lee and Hsu 1992) suggest that the number of relations that need to be expressed can be reduced by decomposing a spatial problem into its constituent dimensions and performing any required reasoning in each dimension in turn. Alternatively, Hernández (1991) proposes a system whereby relationships are described in terms of a *projection* and *orientation* relation, where the first defines how the boundaries of an object relate and the second how areas are placed relative to each other.

However, all such proposals to reduce the number of relations that need to be considered with respect to two-dimensional space are based on the assumption that this space should be represented using a conventional "twin-directional" representation. Using the quad tesseral representation (which has the effect of linearising space) only one-dimensional relationships need to be considered. More specifically the following two relations are suggested:

1. *isSubsetOf* ($\subseteq$): A relation which expresses the requirement that a spatial object is located within some area defined with respect to a second object.
2. *isNotSubsetOf* ($\subseteq$): The negation of the isSubsetOf relation, which expresses the relationship that a spatial object is **not** located within some area defined with respect to a second object.

The area in question can again be defined in terms of a set of tesseral addresses. However, in this case the tesseral addresses are considered to be *offsets* which must be applied to the location set of tesseral addresses associated with the second object. For example given two objects, `X` and `Y` and a 8×8 object space, the relation `Y` within `X` (i.e. that the location set of tesseral addresses `YL` must be a subset of the location set of tesseral addresses `XL`) can be expressed thus:

```
Y isSubsetOf X {000}
```

where the set {000} represents the set of tesseral offsets (D) which must be applied to the location set of tesseral addresses (XL) associated with X.

Similarly the relationship that (say) Y is not to the south-west of X can be expressed as:

```
Y isNotSubsetOf X {3000..3333}
```

where the set of tesseral address {3000..3333} define the south-west quadrant of the example 8×8 object space under consideration. Note that the above actually expresses the relation YL⊆XL+{3000..3333}. Alternatively the above can be considered to express the relation YL−{3000..3333}⊆XL, where D is applied to (tesseraly subtracted from) the set YL instead of XL.

5. CONSTRAINTS AND CONSTRAINT SATISFACTION

Given a means of defining a set of spatial objects and a group of associated relations describing the relationships that exist between these objects, we can use these descriptions to express spatial problems whereby the definition of the set of objects becomes the *domain of discourse* and the relations define the *constraints* affecting this domain. Spatial problem scenarios couched in these terms can thus be categorised as Constraint Satisfaction Problems (CSP). Such problems can generally be defined in terms of a set of variables I = {X1, X2, .., Xn} that can take values from the finite domains {D1, D2, .., Dn} respectively, and a set of constraints which specify which values are compatible with each other. A solution to a CSP is then an assignment of values to all variables which satisfy all constraints. Given that the domains under consideration are finite, and assuming no contradictory constraints, a solution or solutions to a CSP is always possible. The real problem associated with CSP is that of complexity (CSP are NP-complete) and, as a result, efficiency.

The most straightforward method of solving a CSP is to use a "generate and test" paradigm whereby every solution to a CSP is generated and tested against the given constraints. However, this is a very inefficient approach. A more desirable paradigm is to use a depth-first tree search approach supported by some form of *a-priori* pruning so that fruitless branches can be found as early as possible (see van Hentenryck (1989) for further discussion). An alternative approach is to use constraint propagation techniques where variables range over a set of values and constraints are used to restrict this set of values (see Mackworth 1977, Mackworth and Freuder 1985 and Mohr and Henderson 1986). This is the approach that is advocated here. Constraints are used to "prune" the location sets of tesseral addresses associated with each object so that a final configuration (or set of configurations) which satisfies all the constraints is arrived at.

In the previous Section it was proposed that the number of specific relations that need to be considered in two dimensions can be restricted to two, *isSubsetOf* and *isNotSubsetOf*. We express such relations as constraints thus:

```
constraint(X,isSubsetOf,Y,D)
```

or

```
constraint(X,isNotSubsetOf,Y,D)
```

where `D` represents a set of offset tesseral addresses.

The satisfaction of these constraints first requires the generation of the candidate lists of sets of tesseral addresses `XC` and `YC` as described in Section 3. The values `n(XC)` and `n(YC)` then indicate the number of potential branches that will result if the constraint is satisfied with respect to X or Y. Given that we wish to minimise the number of branches in the solution tree each constraint should be satisfied accordingly. Thus the solution of a *isSubsetOf* constraint can be defined as follows:

```
if (n(YC)<(n(XC)) then (∀t∈YC)
      determine the set XLnew = XL∩(t+D)
              if (XLnew = ) no solution for t
      determine the set XCnew
              if (XCnew = ) no solution for t
              otherwise (XLpruned = ⋃(XCnew)) and (YLpruned = t)
otherwise (∀t∈XC)
      determine the set YLnew = YL∩(t-D)
              if (YLnew = ) no solution for t
      determine the set YCnew
              if (YCnew = ) no solution for t
              otherwise (XLpruned = t) and (YLpruned = ⋃(YCnew))
```

where the $\bigcup$ operator indicates the union of all the sets in the indicated list (XC or YC). Note that XCnew and YCnew (the "new" set of candidate locations obtained from XLnew or YLnew as appropriate) are calculated in the same manner, using XS and YS respectively, as that used to calculate XC and YC (see Section 3). Note also that the satisfaction of a constraint results in "pruned" location lists for the objects in question. Note also that the + and - operators used here indicate the tesseral application (addition or subtraction) of every member of the set D to each member of the set t.

The satisfaction of the *isNotSubsetOf* constraint is then defined thus:

```
if (n(YC)<(n(XC)) then (∀t∈YC)
      determine the set XLnew = XL∩(t+D)
              if (XLnew = ) no solution for t
      determine the set XCnew
              if (XCnew = ) no solution for t
              otherwise (XLpruned = union(XCnew)) and (YLpruned = t)
otherwise (∀t∈XC)
      determine the set YLnew = YL∩ (t-D)
              if (YLnew = ) no solution for t
      determine the set XCnew
              if (YCnew = ) no solution for t
              otherwise (XLpruned = t) and (YLpruned = union(YCnew))
```

To demonstrate more clearly the satisfaction of individual constraints, a number of examples are given below and illustrated in Figure 6. The examples clearly illustrate both the versatility of the suggested tesseral approach, and the elegant method whereby constraints defined in this manner can be processed.

EXAMPLE 5 (Figure 6(a)):

Given two tile objects `X` and `Y` such that the `Y` is fixed and `X` is free (with 16 candidate locations), and a requirement that the location of `X` is a subset of `Y` this is expressed as follows:

```
constraint(X,isSubsetOf,Y,{000})
object(X,{000},{000..033})
object(Y,{000},{030})
```

The required processing then produces the result that only one candidate address for `Y` is appropriate. Note that the constraint is resolved with respect to `Y` because `n(YC)` is less than `n(XC)` and that there are no offsets to be applied in this case, i.e. offsets = `{000}`.

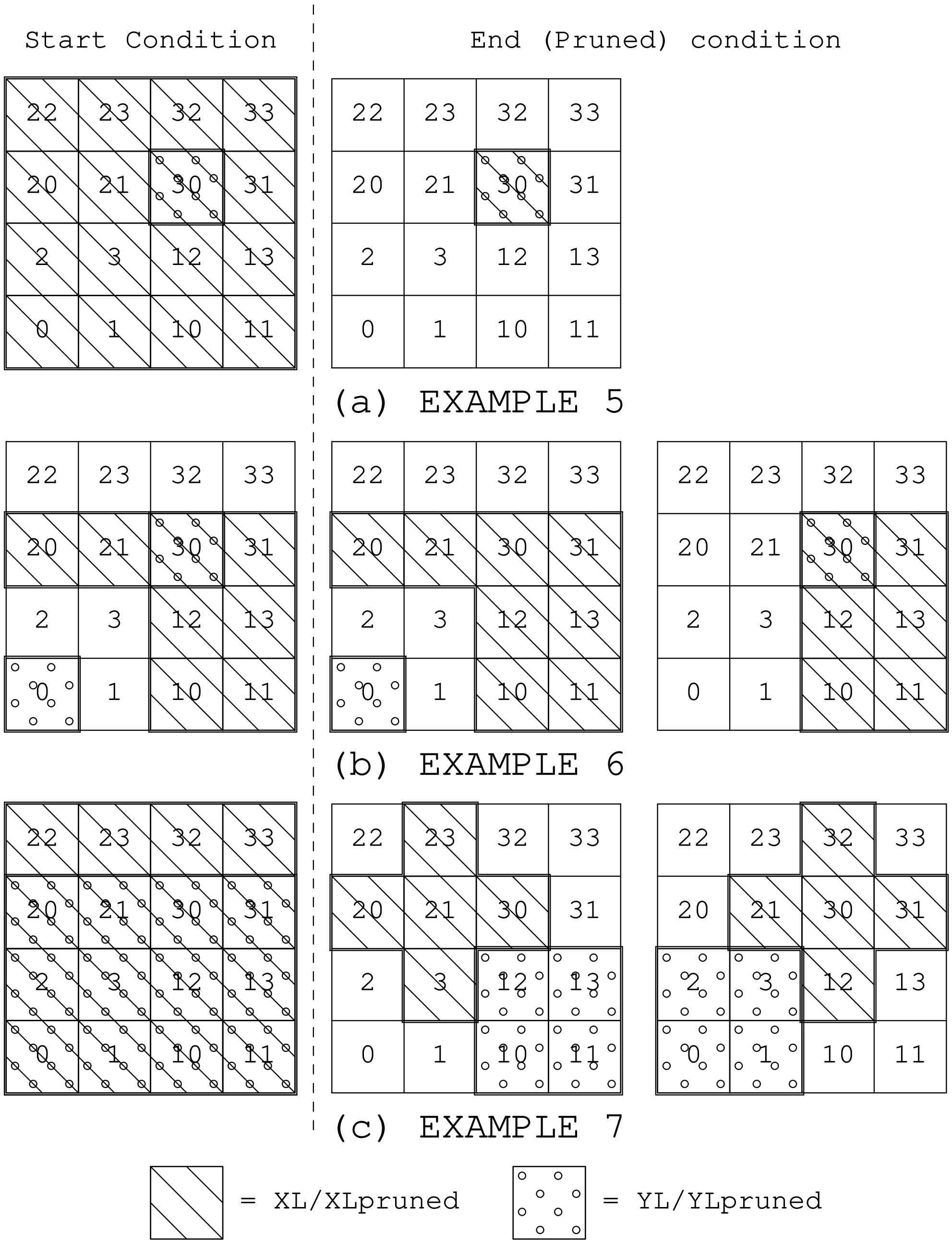


Figure 6: *Constraint satisfaction*

```
n(XC) = 16, n(YC) = 1, thus n(YC)<n(XC)

YC = {{030}}
t  = {030}

XLnew = XL∩(t+D)
      = {000..033}∩{030}+{000}
      = {000..033}∩{030}
```

```
      = {030}

XCnew = {{030}}

XLpruned = {030}, YLpruned = {030}
```

EXAMPLE 6 (Figure 6(b)):

Given a free 2×1 rectangular zone object `X` which has five possible candidate locations, and a free tile object `Y` which has two possible locations, and a requirement that the zone object must not be located immediately to the west of the tile object this is expressed as follows:

```
constraint(X,isNotSubsetOf,Y,{111})
object(X,{000, 001},{010..021, 030..031})
object(Y,{000},{000, 030})
```

The constraint is then processed with respect to `Y` (`n(YC)<n(XC)`). The offset, `{111}` is applied to each candidate location for `Y` and then a new location space for `X` obtained using a "not intersection" operator ($\cap$). This produces two possible new location spaces which satisfy the constraint, and consequently two possible pairs of locations for `X` and `Y`, i.e. the constraint has two possible solutions.

```
n(XC) = 5, n(YC) = 2, thus n(YC)<n(XC)

YC = {{000},{030}}
t  = {000} or {030}

XLnew = XL∩(t+D)

      = {010..021, 030..031}∩{000}+{111} or
      = {010..021, 030..031}∩{030}+{111}

      = {010..021, 030..031}∩{111} or
      = {010..021, 030..031}∩{021}

      = {010..021, 030..031} or
      = {010..020, 030..031}

XCnew = {{010..011}, {012..013}, {020..021}, {021..030},
                         {030..031}} or
      = {{010..011}, {012..013}, {030..031}}

XLpruned = {010..021, 030..031} and YLpruned = {000} or
XLpruned = {010..013, 030..031} and YLpruned = {030}
```

EXAMPLE 7 (Figure 6(c)):

Given two zone objects, X and Y, the first shaped in the form of a 3×3 tile cross and the second in the form of a 2×2 tile box, and the requirement that X must not be contained in Y this will be expressed thus:

```
constraint(X,isNotSubsetOf,Y,{000})
object(X,{001..003, 012, 021},{000..033})
object(Y,{000..003},{000..021, 030..031})
```

The constraint is then satisfied as follows:

```
n(XC) = 4, n(YC) = 6, n(XC)<n(YC)

XC = {{001..003, 012, 021}, {003, 010, 012..013, 030},
     {003, 020..021, 023, 030}, {012, 021, 030..032}}
t  = {001..003, 012, 021} or {003, 010, 012..013, 030} or
     {003, 020..021, 023, 030} or {012, 021, 030..032}

YLnew = YL∩(t-D)

      = {000..021, 030..031}∩{001..003, 012, 021}-{000} or
      = {000..021, 030..031}∩{{003, 010, 012..013, 030}-{000} or
      = {000..021, 030..031}∩{003, 020..021, 023, 030}-{000} or
      = {000..021, 030..031}∩{012, 021, 030..032}-{000}

      = {000, 010..011, 013..020, 030..031} or
      = {000..002, 011, 020..021, 031} or
      = {000..002, 010..013, 031} or
      = {000..011, 013..020}

YCnew =  or
      =  or
      = {{010..013}} or
      = {{000..003}}

XLpruned = {003, 020..021, 023, 030} and YLpruned = {010..013} or
XLpruned = {012, 021, 030..032} and YLpruned = {000..003}
```

5.1. Constraint Selection

The previous section demonstrated how individual constraints can be satisfied by manipulating the associated sets of tesseral addresses using standard set operations. It was also noted that given a set of constraints defining a spatial problem it is desirable that constraints are selected for satisfaction in such a manner that the number of branches in the solution tree are minimised. In the above section it was also illustrated that, given a constraint relating two objects `X` and `Y`, the number of branches that will result from the satisfaction of the constraint will be dependent on the values `n(XC)` and `n(YC)`. Using these values, each constraint making up a spatial problem scenario can be weighted so as to indicate the maximum number of branches that may result on satisfaction of the constraint. Consequently constraints can be selected for satisfaction so as to minimise the number and depth of branches in the solution tree.

6. APPLICATIONS

There are many application areas to which the above spatial reasoning mechanism is well suited. A number of prominent example applications include:

1) Provision of spatial reasoning support for GIS - Geographic Information Systems (Beattie et al. 1995).
2) Environmental impact assessment (Beattie et al. 1996).
3) Timetabling (Coenen et al. 1995).

However, it is conjectured here that the advantages offered through spatial reasoning based on a tesseral representation are applicable to any problem domain which incorporates some spatial element. To provide a greater "feel" for the nature of such applications each of the above listed example applications will be considered in more detail in the following Sub-sections.

6.1. Spatial Reasoning Support for GIS

GIS are a well established part of computer technology and can be found in common usage amongst the public utility providers and local government. However, it is generally acknowledged that such systems have yet to attain their full potential in that they do not currently provide for any spatial reasoning capability concerning the data they model. This is largely due to the Raster and polyline (vector) representations adopted by such systems which are not considered to be conducive to providing for a spatial reasoning capability without resort to computationally expensive techniques. There are many areas where such a capability would offer significant benefits, examples include transportation studies, forest resource management and ship routing. The tesseral representation described here has been found to be an ideal vehicle for providing spatial reasoning support for GIS in that it is directly compatible with raster GIS representations and, by extension, polyline representations. Consequently it can provide all the functionality of traditional GIS while at the same time supporting a spatial reasoning capability.

6.2. Environmental Impact Assessment

Subsequent to the issue of the European Council Directive 85/337/EEC in 1985 all member states of the European Community are required to complete an Environmental Impact Assessment (EIA) report for any major building project. A substantial part of such reports deal with two and three dimensional spatial data and spatio-temporal data. However the computer support currently available is limited to traditional GIS which are not well suited to the required task. To research the requirements for computer support in drawing up EIA, a version of the tesseral reasoning system was used to investigate the noise pollution considerations associated with road building programmes (Beattie et al. 1996). There were two issues to be addressed here, (1) identification of the geographic locations suited to a proposed road programme with respect to given noise pollution levels and (2) identification of noise pollution "zones" that would result along a proposed road plan. It was found that the proposed tesseral system could be used both to model provisional road layouts (and produce noise pollution data reports with respect to those models), and to determine possible "routes" for a desired road system. Further the results were obtained in a computationally inexpensive manner and could be used to produce pictorial displays which could readily be incorporated into EIA documents.

6.3. Timetabling

Timetabling is a well documented Constraint Satisfaction Problem (CSP) and is generally addressed using established CSP techniques. The most common approach is to search through the assignments to be made to find that which is most tightly constrained, and then to make the assignment in a way which constrains subsequent assignments as little as possible. This process continues until a solution is found or the system "gets stuck" in which case previously made assignments must be "swapped". However, it was conjectured (Coenen et al. 1995) that any problem based on a tabular representation is essentially a spatial problem and can be solved using spatial reasoning techniques. A university timetabling application was therefore implemented and tested using the tesseral spatial reasoning technique described here. It was found that the resulting system could easily verify existing timetables by representing all the resources concerned - courses, course units, lecturing slots, rooms and members of staff - as either the axes of the object space or as spatial objects, and defining the required relationships in terms of spatial constraints. It was also shown that by using a multi-dimensional model, two-dimensional "slices" could be produced so that individual timetables could be produced for particular courses or with respect to particular members of staff or rooms. More generally the approach benefited from the broader advantages offered by the representation.

7. EVALUATION

The applications of the tesseral approach to spatial reasoning described above offer both particular and general advantages in each case. However, during the development of the applications it was found that the representation has associated with it the inherent weakness that translation (which requires recourse to look-up tables) was not as computationally efficient as originally envisaged. This has resulted in further work to discover alternative tesseral representations that offer all the advantages of the quad tesseral approach while offering a more computationally efficient techniques to support translation through multi-dimensional space.

A further difficulty encountered during the testing of the above applications was that user errors often occurred when "inputting" address data. This was attributed to the unusual (non-Cartesian) nature of the addressing system and the consequent non-intuitive linearisation that results. This has resulted in further work to develop a tesseral spatial reasoning *scripting* language (supported by a lexical analyser, parser and translator) that will allow users to express spatial problems in a less error prone manner. To this end current further work to identify a more computationally efficient representation is also directed at producing a simpler representation that can be both more readily understood and has a more intuitively obvious linearisation associated with it. In the long term the ideal interface would, of course, be a graphical "point and click" style interface.

A final limitation on the system is that the dimensions of the object space under consideration must have some limit imposed on them. Given a well defined problem, such as a timetabling problem, the dimensions of the object space will be precisely known. Given a less rigorously defined problem, such as an EIA problem, some appropriate limit must be imposed which will serve to meet the problem domain. Whatever the case the object space will always be limited by the maximum integer size supported by the platform on which the system is mounted. Typically, the maximum size of an address will be a 64 bit unsigned integer (some platforms support a 128 bit unsigned integer type). Consequently 2^{64} addresses can be supported allowing definition of a two dimensional object space measuring:

$$\frac{2^{64}}{8} \times \frac{2^{64}}{8}$$

Although it is acknowledged that this maximum object space may not be large enough to address all applications, it is suggested that it is sufficient for a majority of cases.

8. CONCLUSIONS

In this paper we have presented a review of the advantages offered to spatial reasoning when the quad tesseral representation is adopted. The advantages gained (over more traditional multi-dimensional representations) can be summarised as follows:

1) Linearisation of space so that all reasoning need take place only in one-dimension while at the same time supporting the description of complex two-dimensional shapes.
2) Simple description of spatial objects.
3) Effective manipulation of groups of tesseral addresses making full use of (Morton) sequencing and tesseral arithmetic.
4) As a result of 3, computationally efficient satisfaction of the relationships required to exist between spatial objects.

As a result many of the concerns associated with traditional approaches to multi-dimensional reasoning, such as the combinatorial explosion of the number of relationships that need to be considered and the computational expense of manipulating spatial objects, are no longer relevant. Further, by including additional symbols, the approach can be extended to encompass three and four-dimensional reasoning.

The technique, as described in this paper, has been implemented in a tesseral spatial reasoning system which has been successfully used to address a number of applications areas. Both the system and the current versions of the tesseral representations used are still undergoing further development. However, the research team have been greatly encouraged by the results produced so far.

REFERENCES

1. Allen, J.F. (1991). *Time and Time Again: The Many Ways to Represent Time.* International Journal of Intelligent Systems, Vol 6, pp341-355.

2. Beattie, B., 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.

3. Beattie, B., 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.

4. Bell, S.B.M., Diaz, B.M., Holroyd, F.C. and Jackson, M.J.J. (1983). *Spatially Referenced Methods of Processing Raster and Vector Data.* Image and Vision Computing, Vol 1, No 4, pp211-220.

5. Bruce, B.C. (1972). *A model for Temporal References and Application in a Question Answering Program.* Artificial Intelligence, Vol 3, pp1-25.

6. Chang, S.K., Shi, Q.Y., and Yan, C.W. (1987). *Iconic Indexing by 2-D Strings.* IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAM1-9, No 3, May, pp413-428.

7. Coenen, F.P., Beattie, B., 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.

8. Dechter, R., Meiri, I. and Pearl, J. (1991). *Temporal constraint networks.* Artificial Intelligence, Vol 49, pp61-95.

9. Diaz, B.M. and Bell, S.B.M. (1986). *Spatial Data Processing using Tesseral Methods.* Natural Environment Research Council publication, Swindon, England.

10. Freksa, C. (1991). *Qualitative Spatial Reasoning.* In Mark, D.M. and Frank, A.U. (eds). *Cognitive and linguistic Aspects of Geographic Space,* Kluwer, Dordrecht, Netherlands, pp361-372.

11. Freksa, C. (1992). *Temporal Reasoning Based on Semi-Intervals.* Artificial Intelligence, Vol 54, pp199-227.

12. Gargantini, I. (1983). *An effective way to Represent Quadtrees.* Communications of the ACM, Vol 25, No 12, pp905-910.

13. Grunbaum, B. and Shephard, G.C. (1987). *Tilings and Patterns,* Freeman, New York.

14. Hamblin, C.L. (1971). *Instants and Intervals.* Studium Generale, Vol 24, pp127-134.

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

16. Hernández, D. (1991). *Relative Representation of spatial Knowledge: The 2-D Case.* In Mark, D.M. and Frank, A.U. (eds.), *Cognitive and Linguistic Aspects of Geographic Space,* Kluwer, Dordrecht, Netherlands, pp373-385.

17. Holroyd, F.C. (1983). *The Geometry of Tiling Hierarchies.* Ars Combinatoria, Vol 16B, pp211-244.

18. Ladkin, P.B. (1987). *Models of Axioms for Time Intervals.* Proceedings AAAI'87, Seattle, WA, pp234-239.

19. Ladkin, P. (1992). *Effective Solution of Qualitative Interval Constraint Problems.* Artificial Intelligence, Vol 52, pp105-124.

20. Lee, S. Y., and Hsu F. J. (1992). *Spatial Reasoning and Similarity Retrieval of Images Using 2D C-String Knowledge Representation.* Pattern Recognition, Vol 25, No 3, pp305-318.

21. Mackworth, A.K. (1977). *Consistency in Networks of Relations.* AI Journal, Vol 8, No 1, pp99-118.

22. 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.

23. Mohr, R. and Henderson, J.C. (1986). *Arc and Path Consistency Revisited.* Artificial Intelligence, Vol 28, pp225-233.

24. Maiocchi, R. (1992). *Automatic Deduction of Temporal Information.* ACM Transactions on Database Systems, Vol 4, No 4, pp647-688.

25. Morton, G.M. (1966). *A Computer Oriented Geodetic Data Base, and a New Technique in File Sequencing.* IBM Canada Ltd., 1 March 1966.

26. Retz-Schmidt, G. (1988). *Various Views on Spatial Preposition.* AI Magazine, Vol 9, No 2, pp95-105.

27. Samet, H. (1984). *The Quadtree and Related Data Structures.* ACM Comput. Survey, Vol 16, No 2, pp187-260.