TESSERAL SPATIO-TEMPORAL REASONING FOR MULTI-DIMENSIONAL DATA

F.P.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

SUMMARY

A quantitative spatio-temporal reasoning mechanism founded on a tesseral representation of space is described. The principal advantages of this mechanism are that it is universally applicable to any N-dimensional space while at the same time providing for effective manipulation and comparison of objects located within such spaces. These advantages are a direct consequence of the derived tesseral representation on which the mechanism is founded. The defining features of this representation include:

- Efficient translation through N-dimensional space using conventional integer arithmetic.
- Computationally effective rotation.
- Linearisation of space providing for the application of one-dimensional (temporal) reasoning techniques regardless of the number of dimensions under consideration.
- Run line encoding.
- Simple relational comparison of N-dimensional objects.

In addition the representation may be viewed as either an intermediate representation which links the reasoning mechanism with some primary representation (raster and by extension vector), or as

a primary representation in its own right.

The mechanism is incorporated into a demonstration spatio-temporal reasoning system, the SPARTA (SPAtial Reasoning using Tesseral Addressing) system. This currently operates using a constraint satisfaction approach to spatial problem solving supported by a heuristically guided constraint selection mechanism so as to minimise the search space. Spatial problems are passed to this system in the form of a script comprising a set of object descriptions (defined in terms of classes and instances of those classes) and a set of constraints defining the relationships "desired" to exist between pairs or groups of objects. Successful resolution of a script results in the production all solutions that satisfy the given constraints (assuming a solution exists). The resulting output can be either graphical or textual as directed by the user.

The approach has been applied to a number of practical application areas including; site identification for civil engineering projects, electronic chart interaction and noise pollution analysis. Other, more esoteric, applications that have been investigated include classic AI problems such as N-queens problems, and multi-dimensional shape fitting scenarios.

The paper gives a detailed description of the representation and an overview of the SPARTA N-dimensional reasoning system and its associated constraint satisfaction mechanism. The paper is illustrated with a sequence of shape fitting scenarios, ranging from 1 to 4-dimensions, which demonstrate fully the advantages of the representation and that the proposed approach is applicable to any number of dimensions without requiring alterations to the representation or the supporting reasoning mechanisms.

-- --

TESSERAL SPATIO-TEMPORAL REASONING FOR MULTI-DIMENSIONAL DATA

F.P. 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

A versatile and universally applicable quantitative multi-dimensional reasoning mechanism founded on a unique linear tesseral representation is described. A consequence of this linearity is that quantitative multi-dimensional spatial reasoning can be implemented in one-dimensional terms. Compatibility with raster data renders the mechanism generally applicable to a wide range of applications, examples include spatial reasoning for Geographic Information Systems (GIS), electronic chart and map interaction, and image recognition. In addition the representation has an arithmetic which provides translation and rotation through N-dimensional space using standard integer arithmetic. The mechanism has been incorporated into a spatio-temporal reasoning system, the SPARTA (SPAtial Reasoning using Tesseral Addressing) system, which has been applied successfully to a diverse number of spatial applications.

Keywords: Quantitative Reasoning, Spatio-Temporal Reasoning, Tesseral Addressing, N-Dimensional Data.

1. INTRODUCTION

There are many applications where a reasoning capability applicable to raster data is of significant benefit. In this paper we consider a quantitative spatio-temporal reasoning mechanism founded on a tesseral representation of space which is universally applicable to N-dimensional spatial data stored or presented using a raster encoding. Example raster representations include those used for image encodings (GIF, PBM etc.), Geographic Information System (GIS) data storage and 3-D visualisation. Applications that have been investigated by the authors include; site identification for civil engineering projects, electronic chart interaction and noise pollution determination (see Beattie et al.[1, 2] and Coenen et al.[3] for further detail). Other, more esoteric, applications include classic AI problems such as N-queens problems, and multi-dimensional shape fitting scenarios. The latter are used to illustrate this paper.

The universal applicability offered by the reasoning mechanism is a direct consequence of the derived tesseral representation on which it is founded. This representation may be viewed as either an intermediate representation which links the reasoning mechanism with some primary representation (raster and by extension vector), or as a primary representation in its own right.

The tesseral reasoning mechanism described is founded on a constraint satisfaction approach to spatial problem solving supported by a heuristically guided constraint selection mechanism so as to minimise the search space. The mechanism has been incorporated into a demonstration spatio-temporal reasoning system, the SPARTA (SPAtial Reasoning using Tesseral Addressing) system. Spatial problems are passed to this system in the form of a script comprising a set of object descriptions (defined in terms of classes and instances of those classes) and a set of constraints defining the relationships "desired" to exist between pairs or groups of objects. The system then produces all solutions that satisfy the given constraints and outputs these solutions, either in some graphical or textual format, as directed by the user.

More detailed overviews of the proposed tesseral representation and the SPARTA system are presented in sections 2 and 3 respectively. The operation of the system is described in sections 4, 5, 6 and 7 in terms of 1-, 2-, 3- and 4-dimensions respectively. Each of these sections is illustrated with a shape fitting scenario. Finally, brief notes concerning the constraint satisfaction mechanism are given in section 8, and some conclusions in section 9.

2. OVERVIEW OF REPRESENTATION

The theory of tesseral representation is well established (Bell et al.[4] , Diaz and Bell[5], Gargantini[6]). Conceptually tesseral representations are founded on ideas concerning the hierarchical decomposition of N-Dimensional space into isohedral sub-spaces until some application dependent resolution is reached (Grunbaum and Shephard[7]). Each sub-space is then referenced using some symbolism which serves to linearise the space. Typically a number system is chosen such that an arithmetic can be applied to the addresses (e.g. Holroyd[8]).

In the case of the proposed representations, given an *object space* (i.e. a space within which one or more objects of interest can be said to exist), this space is conceptually tessellated down to some predefined resolution in a manner similar to the decomposition associated with linear quad-tesseral methods. Each resulting sub-space is then given a reference (address) comprising a single integer whose bit pattern is made up of a set of coordinates. For the purpose of this document we will assume a 32 bit signed integer with a bit pattern 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 (it is not necessary to include groupings for dimensions which are not relevant to a particular application).
3) Similarly, it is not necessary to allocate identical numbers of bits to each dimension.
4) The sign bit is included to provide for negative addressing required in order to support various translation operations.

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

Given the above, addresses can be calculated using the identity:

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

The reverse conversion can be achieved using appropriate sequences of left and right shifts to isolate sequences of bits.

The principal advantages offered by the above tesseral representation over other tesseral forms, such as the linear quad-tesseral (Gargantini[6]) and hierarchical quad-tesseral (Samet[9]) representations, are as follows:

1) The resulting "bottom-left to top-right" linearisation (see Figure 3 for a example) is intuitively obvious and, more importantly, predictable (i.e. given any sub-space the addresses of all the physically adjacent sub-spaces can always be predicted). Traditional representations typically have a "Z" linearisation associated with them, often referred to as a Morton linearisation (after Morton[10]). Although this linearisation offers advantages of data clustering, it is counter-intuitive and non-predictable (we cannot predict the addresses of physically adjacent spaces to a given sub-space).
2) Translation and rotation algorithms are much more computationally efficient (assuming standard integer processors). To be effective traditional tesseral arithmetics required specialised tesseral processors. In the (likely) absence of such processors the arithmetic becomes computationally expensive, to the extent that it is cheaper to convert addresses into a Cartesian form, perform the required arithmetic on the resulting coordinates, and then to reconvert to addresses.

The principal disadvantage of the representation is that the degree of data compression is not, on occasion, as good as that achieved by other tesseral representations, although the difference is minimal and is far offset by the advantages of efficient translation and rotation offered by the approach. An additional feature offered by quad-tesseral and hierarchical tesseral representations is that of data clustering. This is of particular significance for applications where data access time is important; for example where the primary objective is to access and display data as in the case of standard GIS applications. However, with respect to the reasoning mechanism described here, where (a) the representation is viewed as an intermediate representation and (b) data display is not an objective, it can reasonably be assumed that data access is an issue for the primary representation before conversion to the intermediate. In circumstances where the representation constitutes the primary representation, and there are many applications where this is a viable option, any access time limitations will again be offset by the advantages of efficient translation and rotation offered by the approach.

3. OVERVIEW OF SPARTA

The SPARTA system recognises a number of categories of object. In the context of this paper the following are significant:

1) Fixed objects which cannot be moved, i.e. they have a "fixed" location.

2) Free objects which have no fixed location (and thus can be moved).

Objects are defined in terms of classes and instances of classes. A class comprises a name and a category definition (e.g. `fixed` or `free`). In the case of a free object the class definition includes a shape definition described in terms of a set of addresses. Instance definitions then create instances of classes and define either the location of the instance or the location space within which the instance can be said to exist. The first is applicable to fixed objects and the second to free objects. Constraints are then used to define the relationships that are desired to exist between object locations.

Classes, instances and constraints are expressed in a specially developed scripting language. Syntactically this language comprises four primary constructs on which all SPARTA scripts are founded:

```
space(MaxCoordinates).

class(Class_Name,Object_Category,Shape_Definition).

instance(Instance_Name,Class_Name,Location_Definition).

constraint(Locations1,Relation,Locations2).
```

The first defines the desired object space in terms of its maximum coordinates. Shape and location definitions are defined in terms of sets of addresses. With respect to the last construct particular locations are identified through the use of instance or class names. To facilitate script writing, there are various defaults and other short cuts that can be used, some of which will be illustrated later in this document. To address a number of more taxing scenarios, various refinements to the above have also been introduced (an example is introduced in section 4). The complete ontology used to develop the SPARTA scripting language can be found in Coenen et al.[11].

From the foregoing it can be seen that a constraint is processed with respect to two sets of locations. On satisfaction of the constraints there may be 0, 1 or more solutions, i.e. groups of compatible locations. If there are 0 solutions the constraint is said to fail indicating that the scenario cannot be physically realised, otherwise the constraint is said to succeed. Where there is only one possible solution the objects in question become fixed objects - they have become associated with a fixed location. Where there is more than one solution at least one of the objects remains a free object. For example given two instances, `P` which can be located at `[a, b, c]`, and `Q` which can be located at `[b, c, d]` (where `a`, `b`, `c` and `d` are sets of addresses each representing a possible location for `P` and/or `Q`), and a constraint:

```
constraint(LocationsP, notEqual, LocationsQ).
```

three groups of compatible solutions, that satisfy the constraint, would be produced:

	Locations for P	Locations for Q
Solution1	[a]	[b, c, d]
Solution2	[b]	[c, d]
Solution3	[c]	[b, d]

Note that the maximum number of valid pairings of locations, given any two groups of locations, is equivalent to the size of the Cartesian product of the two groups.

4. ONE-DIMENSIONAL (TEMPORAL) TESSERAL REASONING

A substantial and well founded literature exists concerning temporal reasoning (frequently quoted work includes Allen[12], Freksa[13], Jixin and Knight[14], Kowalski and Sergot[15], and Vilain[16]). Temporal reasoning systems tend to be either qualitative (e.g. van Beek[17]) or quantitative (e.g. Bruce[18]), the distinction is that the latter is referenced to a *number line* of some description. Typically, temporal objects are represented using points or intervals (or both) which are related through relational descriptors (e.g. `before`, `after`, `equals` etc.). Descriptions can be manipulated using *predicate logic* (e.g. Kowalski and Sergot[15], McDermott[19], Shoham[20]), *transitivity table* (Allan[12]) or *constraint satisfaction* (Dechter et al.[21], Ladkin[22]) techniques to arrive at further (qualitative) descriptions linking temporal objects and/or (quantitative) locations of objects along the number line.

Consider a one dimensional space comprising 20 points with (X) coordinates ranging from 0 to 19 inclusive. Using the proposed tesseral representation this will comprise a simple number line consisting of 20 reference addresses as demonstrated in Figure 1.

*** **Figure 1:** *One-dimensional object space* ***

Inspection of Figure 1 demonstrates many of the advantages/features associated with the proposed representation.

1) **Linearisations**: The space is presented as a (subset of a) continuous linear sequence of addresses.
2) **Simple translation**: Movement through the space can be achieved through straightforward integer addition.
3) **Run line encoding**: Sequences of reference addresses related to particular objects can be stored in terms of their start and end addresses. With respect to this paper such sequences will be indicated using a '..' infix operator, where the prefix operand represents the start address, and the postfix operand the end address.
4) **Simple comparison of sets of addresses**: To determine whether one set of addresses stands in some relation to another set of addresses it is not necessary to compare each address in one with each address in the other. Instead, knowledge of the relative location within the number line of sequences of addresses can be used to achieve more efficient comparison of sets of addresses.

5) **Simple addition (and subtraction) of sequences of addresses**: When adding (or subtracting) one sequence of addresses to another it is only necessary to add (or subtract) the start and end addresses.

6) **Simple and computationally effective rotation**: Any object can be rotated (in one dimensional terms) by applying the reference `-1` to its shape definition.

With respect to the SPARTA scripting language a space of the form shown in Figure 1 would be defined as follows:

```
space([19]).
```

To illustrate the operation of the reasoning mechanisms in one dimension consider the 1-D objects presented in Figure 2(a), three of which (`b1`, `b2` and `c1`) are discontinuous. Using the SPARTA scripting language the nature of these objects can be defined as a group of classes, for example:

```
class(classA, free, [0..4]).
class(classB, free, [0..3, 9..10]).
class(classC, free, [0, 3..4]).
```

Note that in this example the objects are assumed to be free objects (they have no fixed location), and that the shape of each object is defined using sets of tesseral addresses (the nature of the shapes can be verified by reference to Figure 2(a)). A number of instances of these classes can then be defined as follows:

```
instance(a1, classA).
instance(b1, classB).
instance(b2, classB).
instance(c1, classC).
```

Note that by default the location space for free objects is assumed to encompass the entire object space (`[0..19]` in this case). Now suppose that these instances were required to be fitted into the defined object space in such a manner that no one instance overlaps any other. This could be expressed in terms of the following group of constraints:

```
constraint(a1, complement, b1).
constraint(a1, complement, b2).
constraint(a1, complement, c1).
constraint(b1, complement, b2).
constraint(b1, complement, c1).
constraint(b2, complement, c1).
```

However, the scripting language provides a more convenient shorthand method of defining such constraints:

```
constraint([a1, b1, b2, c1], complement).
```

To determine a solution to the above the script is passed to the SPARTA system and, on completion of the constraint satisfaction mechanism, a solution produced. In the above case this will result in the eight solutions presented in Figure 2(b).

*** **Figure 2:** *One-dimensional shape fitting scenario* ***

One dimensional shape fitting scenarios of the above form have many similarities with scheduling problems where a number of temporally dimensioned tasks must be fitted into a time line (see Coenen et al.[23] for further discussion), although, in that case rotation must be prevented. Further, it is also usually necessary to specify that certain tasks must be completed before other tasks commence. For example, in the above script, if we wished to insist that `a1` is completed prior to the start of `c1` we would include a constraint of the form:

```
constraint(a1, within, (c1, offset(ref,[-1..-19]))).
```

Note that the SPARTA scripting language syntax does not support a "before" relation as this implies a direction which, although meaningful in 1-D, becomes difficult to define in higher diemensions. The above constraint will be satisfied only by those locations associated with `a1` which are "within" the space defined by the locations for `c1` with the *offset* `[-1..-19]` applied, i.e. the space towards the zero address (the origin) of any specific location for `c1`. The offset in this case is applied only to the reference addresses for the `c1` locations. This reference address, in turn, is defined as the address of the minimum bounding box for each location which is nearest to the origin. The consequence of incorporating this constraint into the above script would be to preclude the last four solutions presented in Figure 2.

The offset concept is discussed further in the following section. Additional information regarding the operation of the SPARTA system in one dimensional terms can be found in Coenen et al.[24, 25].

5. TWO-DIMENSIONAL (SPATIAL) TESSERAL REASONING

In the previous section the use of the proposed representation and its associated reasoning mechanism was discussed with respect to one-dimensional problem solving. From the above it can be seen that in many respects the SPARTA system is not unlike many other quantitative temporal reasoning systems. However, the real strength of the approach is that all the advantages offered in one-dimension are equally available in higher dimensions without the need for any adjustments or changes to the representation or the reasoning mechanism. Consider the following object space definition:

```
space([4,4]).
```

This defines an object space of the form illustrated in Figure 3.

*** **Figure 3:** *Two-dimensional object space* ***

Note that:

1) The representation again causes the space to be linearised, thus two-dimensional reasoning can be implemented in one-dimensional terms.
2) Translation, in any direction, is still achieved through simple integer addition and subtraction (e.g. +1 to move in the X direction and +256 to move in the Y direction).
3) The representation still supports runline encoding and simple comparison and addition/subtraction of sets of addresses.
4) The representation is analogous with raster image and GIS representations, thus providing compatibility with such representations.

5) Orthogonal rotation is implemented as a special case of the standard approach taken from complex number theory (see, for example, Jeffrey[26]for further discussion):

$$\mathtt{T_a \times T_b = (x_a x_b - y_a y_{b)} + Base\,(x_a y_b + y_a x_{b)}}$$

where $\mathtt{T_a}$ and $\mathtt{T_b}$ are two tesseral addresses, and $\mathtt{x_a}$, $\mathtt{x_b}$, $\mathtt{y_a}$ and $\mathtt{y_b}$ represent the $\mathtt{x}$ and $\mathtt{y}$ coordinates associated with $\mathtt{T_a}$ and $\mathtt{T_b}$ respectively. Recall that the 2-D `base`, within the context of this paper, is equivalent to `256`.

Thus to rotate some set of addresses through 180° we still apply the reference `-1`, however to rotate through 90° we would apply the reference `256`:

$$\mathtt{T_a \times 256 = 256\,(x_{a)} - y_a}$$

The implementation of the above necessitates conversions to sets of coordinates, but not reconversion, as the result produced by the identity will be an address. However, given the simplicity of the conversion algorithm (see section 2), this is not considered to represent a significant drawback when the other advantages offered by the representation are taken into consideration.

Some additional discussion concerning the problems associated with the scaling up of 1-D reasoning techniques to address 2-D problems is presented in the following sub-section. The application of the SPARTA system, in terms of its two-dimensional operation, is then described in sub-section 5.2.

5.1. Scaling up from one to two dimensions

The most obvious approach to implementing multi-dimensional spatial reasoning is to simply scale up tried and tested one-dimensional (temporal) techniques. However, experience shows that in attempting this it quickly becomes apparent that the number of relations to be considered increases exponentially with the number of dimensions under consideration (Retz-Schmidt[27]). In qualitative systems we can identify 3 point-point relations, 5 point-interval relations, 5 interval-point relations and 13 interval-interval relations, giving a total of 26 (Hamblin[28]). Some authors make a distinction between `before` and `one-before`, and `after` and `one-after`, giving a total of 34 relations (Jixin and Knight[14]). Assuming the latter, scaling up to 2-, 3- and 4-D respectively gives:

Dimension	Total number of relations
2-D	$(5^2) + (7^2) + (7^2) + (15^2) = 348$
3-D	$(5^3) + (7^3) + (7^3) + (15^3) = 4186$
4-D	$(5^4) + (7^4) + (7^4) + (15^4) = 56052$

Note that the above totals do not include the negation of relations or the inclusion of partial relations such as those proposed by Freksa[13].

Much current work on spatial reasoning is directed at alternative approaches to defining qualitative spatial relations. The most obvious approach is to consider individual dimensions independently and reason about each in isolation (e.g. Malik and Binford[29], or Chang et al.[30]). Alternatively we can restrict the number of relations to be considered to topological relations only (e.g. Cui et al.[31]) and develop techniques to effectively manipulate such relations, such as Egenhofer's 9-intersections approach (Egenhofer[32]) or higher order logics (Cohn et al.[33]). A further alternative is to develop systems that support different levels of relation, such as the projection and orientation relations suggested by

Hernández[34].

The effect of the linearisation associated with the SPARTA system renders many of the above concerns to be of no consequence in that all reasoning is implemented in one-dimensional terms. Constraints define relationships between locations described in terms of sets of addresses, i.e. constraints are expressed in terms of set relations of which there are only a small number:

$=$	Equals		Complement
$\subset$	Proper Subset	$\subseteq$	Subset
$\supset$	Proper Superset	$\supseteq$	Superset
$\cap$	Intersection	$\cup$	Union

Including the negation of the above, this gives a total of 16 relations. Further, the authors have not found a scenario where the distinction between Subset and Proper-Subset, and Superset and Proper-Superset, are relevant. Thus the practical number of relations that needs to be considered is 12. However, with respect to SPARTA scripts, there is still a need to be able to express all possible relations (both qualitative and quantitative) that can exist between spatial objects. Using the above 12 relations many of the standard spatial relations (within, outside, overlapping etc.) can be expressed. The remaining standard relations (northEastOf, externallyTangential etc.), and more specific quantitative relations, can then be expressed by applying an *offset* to the locations associated with an object (an example was given in section 4). Such offsets comprise further sets of addresses and are applied prior to the solution of the constraint in question.

5.2. 2-D Shape Fitting Scenario

Given the 2-D object space presented in Figure 3 and the shapes presented in Figure 4(a) we can define a SPARTA script as follows which will determine all possible configurations whereby the given shapes can be fitted into the 2-D object space.

```
space([4,4]).

class(classA, free, [0, 256..257, 512]).
class(classB, free, [0..2, 256, 512]).
class(classC, free, [0, 256, 512]).
class(classD, free, [0..1,, 257..258, 514]).
class(classE, free, [2..3, 256..258]).
class(classF, free, [0..1, 256]).

instance(a1, classA, noRotation).
instance(b1, classB).
instance(c1, classC).
instance(d1, classD).
instance(e1, classE).
instance(f1, classF).

constraint([a1,b1,c1,d1,e1,f1], complement).
```

Note that, so as to prevent the generation of multiple views of each solution, the above script insists that one of the instances (`a1`) is not rotated. The result is presented in Figure 4(b).

*** **Figure 4:** *Two-dimensional shape fitting scenario* ***

Two-dimensional shape fitting scenarios of the above form are analogous to floor plan design problems where a number of objects must be fitted into a given area (e.g. a factory floor), and to 2-D civil engineering site location applications.

6. THREE-DIMENSIONAL (SPATIAL) TESSERAL REASONING

All the advantages offered by the proposed reasoning mechanism as applied to 2-D, can similarly be extended to encompass 3-D reasoning, again without the need for any changes to the system. Using the proposed representation, and given a 3-D object space, a notional 3-D grid will be produced. Thus an object space measuring $3 \times 3 \times 3$ would provide a grid of the form shown in Figure 5.

*** **Figure 5:** *Three-dimensional object space* ***

Again the space is linearised and thus the same mechanism can be used to process scripts as was used in the case of the 1-D and 2-D scenarios illustrated above. Similarly the advantages of translation, rotation, runline encoding, simple comparison and addition/subtraction of sets of addresses are also applicable.

The following script defines a 3-D shape fitting problem where the five shapes defined in Figure 6(a) are fitted into the given object space. Again, so as to prevent the generation of 24 different views of identical results, rotation is prevented for one of the instances. The two possible solutions are presented in Figure 6(b).

```
space([2,2,2])

class(classA, free, [0, 256..257, 513..514, 66049]).
class(classB, free, [0, 255..256, 512, 66048..66049]).
class(classC, free, [0..1, 256..257, 65536]).
class(classD, free, [0..1, 257, 513, 65537, 66049]).
class(classE, free, [0..1, 65536, 131072]).

instance(a1, classA, noRotation).
instance(b1, classB).
instance(c1, classC).
instance(d1, classD).
instance(e1, classE).

constraint([a1, b1, c1, d1, e1], complement).
```

*** **Figure 6:** *Three-dimensional shape fitting scenario* ***

Three-dimensional shape fitting problems of the above form are analogous to 3-D site location problems, such as those found in environmental impact assessment and civil engineering projects, where a number of objects must be fitted into a proposed development area so as as to minimise the environmental impact and maximise utility (see Coenen et al.[3] for further discussion). Such applications often require that certain objects are located a given distance from other objects. Where this is a requirement, for example as a consequence of noise pollution concerns, a constraint of the following form may be included:

```
constraint(a1, complement, (b1, offset(sphere(5)))).
```

This states that locations for the instance a1 are only valid if they do not intersect the location of instance b expanded in all directions by 5 tiles. By default offsets are applied to all addresses associated with their target objects.

7. FOUR-DIMENSIONAL (SPATIAL) TESSERAL REASONING

The advantages offered by the proposed representation and reasoning mechanism are equally applicable to 4-D space. Again translation is achieved through simple integer addition and subtraction, object data can be stored using runline encoding and manipulated effectively through knowledge of the linearisation. Rotation can be achieved in a similar manner to that described for 2-D and 3-D applications as described in the previous two sections. However, the precise meaning of rotation through 4-D is beyond the scope of this paper.

In Figure 7 a $3\times3\times3\times3$ object space is presented in the form of a series of slices through the fourth dimension which, in the context of this paper, is assumed to be time (T).

*** **Figure 7:** *Four-dimensional object space* ***

In Figure 8(a) a number of 3-D objects are presented, a1, b1, c1 and c2. In the following script it is assumed that object a1 exists for 3 units of time, object b1 for two units of time and objects c1 and c2 for one unit of time each, and that these 4-D objects are required to be fitted into the object space presented in Figure 7 without overlap. The result is presented in Figure 8(b).

```
space(2,2,2,2).

class(A, free, [0..2, 256..258, 512..514, 65536, 65792,
    66048..66050, 131072, 131328, 131584..131586,
    16777216..16777218, 16777472..16777474, 16777728..16777730,
    16842752, 16843008, 16843264..16843266, 16908288, 16908544,
    16908800..16908802, 33554432..33554434, 33554688..33554690,
    33554944..33554946 33619968, 33620224, 33620480..33620482,
    33685504, 33685760, 33686016..33686018]).
class(B, free, [0..1, 256..257, 65536..65537, 65792..65793,
    16777216..16777217, 16777472..16777473 16842752..16842753,
    16843008..16843009]).
class(C, free, [0..1, 256..257])

instance(a1, A, noRotation).
instance(b1, B).
instance([c1,c2], C).

constraint([a1,b1,c1,c2], complement).
```

*** **Figure 8:** *Four-dimensional shape fitting scenario* ***

Shape fitting scenarios of the above form are analogous to problems associated with stowage plans for mixed goods carried in cargo containers or railway wagons where particular goods are to be unloaded/loaded at discrete time points during the container's transit between two termini. Of course, to ensure that goods to be unloaded are accessible at appropriate points, further constraints must be added. The following constraint ensures that c1 is not overloaded by any other object (assuming the container is only open at the top):

```
constraint([a1,c2,b1], complement, c1, offset([65536, 65792])).
```

Again use is made of a tesseral offset to describe the desired restriction. It would also be necessary to specify more precisely the life span of instances from the point in time when they are loaded to the point in time when they are discharged. In other words the location space during which the instances exist must be redefined (the above script assumes, by default, that this is equivalent to the object space). A suitable redefinition might be as follows:

```
instance(a1, A, noRotation).
instance(b1,B, [box 0 16908802]).
instance([c1,c2],C, [box 0 131586]).
```

Thus a1 still exists from T=0 to T=2 (by default), b1 from T=0 to T=1, and c1 and c2 during T=2. The effect of incorporating these additions/changes into the above script is to preclude all but the first five (S1, S2, S3, S4 and S5) solutions presented in Figure 8.

8. NOTE ON CONSTRAINT SATISFACTION MECHANISM

There are many computer applications where a solution is required to be generated which conforms to a set of constraints. Examples include: machine vision, belief maintenance, scheduling and time-tabling, graph colouring, logic puzzles, floor plan design and circuit design. This class of applications is often referred to as Constraint Satisfaction Problems

(CSP) (van Hentenryck[35]). Temporal and spatial reasoning problems also fall into the category of CSP. More formally a CSP can be defined in terms of a set of variables `I = {X1, X2, .., Xn}` which 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. Thus a constraint of the form `c(Xi1, Xi2, .., Xik)` between `k` variables from `I` is a subset of the Cartesian product `Di1 x Di2 x Dik`. A solution to a CSP is then an assignment of values to all variables which satisfy all constraints. Given that the domains `{D1, D2, .., Dn}` are finite, a solution or solutions to a CSP can always be found provided that constraints are not contradictory. The real problems associated with CSP are that of complexity (CSP are *NP-complete*) and, as a result, efficiency.

The solution to CSP is typically founded on some tree searching strategy supported by a mechanism whereby fruitless branches within the tree can be identified early in the search process. Examples included dependency based backtracking and *a priori* pruning techniques such as forward checking and "look ahead" (see Mackworth[36] for further discussion). Given a particular CSP the nature of the problem can be further exploited to achieve additional pruning by guiding the search using application dependent heuristics. This was the idea behind a number of problem solving systems such as ALICE (Lauriere[37]).

CSP tree searching strategies are well suited to encoding using logic programming languages (e.g. PROLOG). However, the current disadvantage of such languages is that their execution is very inefficient. Search procedures (i.e. backtracking) are based on recovery from failure rather than avoiding failure. Consequently a number of constraint (logic) programming languages have been developed to address the disadvantages associated with standard logic languages. One example is CHIP (Constraint Handling in PROLOG) which extends standard PROLOG to include the concepts of finite domains, boolean terms and relational terms (see Dincbas et al.[38]). Further examples include: PROLOG III (Colmerauer[39]), and CLP (Constraint Logic Programming) which is a framework for constraint handling in logic (see Jaffar and Michaylov[40]), which in turn is an extension of the earlier logic programming language scheme (Jaffar et al.[41]). Alternatively, many researchers have resorted to implementing CSP solution strategies using more traditional imperative programming languages. The current version of the SPARTA system is written in C.

The design of the SPARTA system is such that the constraint satisfaction mechanism is independent of the representation, consequently the system is not tied to any particular approach to CSP solving. However, the current versions of the system use a heuristically guided depth first search strategy. Partial and end solutions are stored in a solution tree which is constructed (and destructed) dynamically as the solution process progresses. Each node in the tree 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. Consequently there are two efficiency concerns:

1) To limit the growth (in width) 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 on in the solution process as possible.

These two concerns are linked in that the early identification of fruitless branches in the solution tree will also have the effect of limiting the growth of the tree. To achieve the above the following pruning techniques 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. one 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 (and consequently the number of nodes in the tree).

9. CONCLUSIONS

A qualitative spatio-temporal reasoning mechanism has been described founded on a tesseral representation and linearisation of space. The mechanism offers the following significant advantages:

1) It is universally applicable regardless of the number of dimensions under consideration.

2) It is fully compatible with Raster representations rendering it suited to a wide range of applications founded on such representations, e.g. image analysis, reasoning for GIS and map interaction. Experiments have also indicated that the technique can be used for applications which are not normally founded on spatio-temporal representations such as scheduling and timetabling.

3) It is conceptually simple and computationally effective.

These advantages are gained primarily as a consequence of the particular tesseral representation on which the mechanism is founded.

The proposed mechanism has been incorporated into a spatio-temporal reasoning system, the SPARTA system, which has been tested against a great many spatial problems, a number of which were referred to in this paper. Current work is focused on noise pollution modelling and assessment in the city of London, and marine electronic chart interaction.

References

1. B. Beattie, F.P. Coenen, T.J.M. Bench-Capon, B.M. Diaz and M.J.R. Shave, Spatial Reasoning for GIS using a Tesseral Data Representation, in Revell, N. and Tjoa, A.M. (Eds), *Database and Expert Systems Applications* (Proceedings DEXA'95), Lecture Notes in Computer Science 978, Springer Verlag, 207-216 (1995).

2. B. Beattie, F.P. Coenen, A. Hough, T.J.M. Bench-Capon, B.M. Diaz and M.J.R. Shave, 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 (1996).

3. F.P. Coenen, B. Beattie, T.J.M. Bench-Capon, B.M. Diaz and M.J.R. Shave, Spatial reasoning for geographic information systems. *Proceedings 1st International Conference on GeoComputation*, School of Geography, University of Leeds **1** 121-131 (1996).

4. S.B.M. Bell, B.M. Diaz, F.C. Holroyd and M.J.J. Jackson, Spatially referenced methods of processing raster and vector data, *Image and Vision Computing* **1**(4) 211-220 (1983).

5. B.M. Diaz and S.B.M. Bell, Spatial data processing using tesseral methods, *Natural Environment Research Council publication*, Swindon, England (1986).

6. I. Gargantini, Linear octrees for fast processing of three dimensional objects. *Computer Graphics and Image Processing* **20** 365-374 (1982).

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

8. F.C. Holroyd, The Geometry of tiling hierarchies, *Ars Combinatoria* **16B** 211-244 (1983).

9. H. Samet, The Quadtree and Related Data Structures, *ACM Comput. Survey* **16**(2)187-260 (1984).

10. G.M. Morton, A computer oriented geodetic data base, and a new technique in file sequencing, *IBM Canada Ltd.* (1966).

11. F.P. Coenen, B. Beattie, T.J.M. Bench-Capon, M.J.R. Shave and B.M. Diaz, An ontology for linear spatial reasoning, in R.R. Wagner and H. Thomas (Eds), *Database and Expert Systems Applications, (Proceedings DEXA'95)*, Lecture Notes in Computer Science 1134, Springer Verlag, 718-727 (1996).

12. J.F. Allen, Maintaining knowledge about temporal intervals, *Communications of the ACM* **26**(11) 832-843 (1983).

13. C. Freksa, Temporal reasoning based on semi-intervals, *Artificial Intelligence* **54** 199-227 (1992).

14. M. Jixin and B. Knight, A general temporal theory, *The Computer Journal* **37**(2) 114-123 (1994).

15. R. Kowalski and M. Sergot, A logic-based calculus of events, *New Generation Computing* **4** 67-95 (1986).

16. M.B. Vilain, A system for reasoning about time, *Proceedings AAAI'82* 197-201 (1982).

17. P. van Beek, Reasoning about qualitative temporal information. *Artificial Intelligence* **58** 297-326 (1992).

18. B.C. Bruce, A model for temporal references and application in a question answering program, *Artificial Intelligence* **3** 1-25 (1972).

19. D. McDermott, A temporal Logic for reasoning about processes and plans, *Cognitive Science* **6** 101-155 (1982).

20. Y. Shoham, Temporal logics in AI: semantical and ontological considerations, *Artificial Intelligence* **33** 89-104 (1987).

21. R. Dechter, I. Meiri and J. Pearl, Temporal constraint networks, *Artificial Intelligence* **49** 61-95 (1991).

22. P. Ladkin, Effective solution of qualitative interval constraint problems, *Artificial Intelligence* **52** 105-124 (1992).

23. F.P. Coenen, B. Beattie, T.J.M. Bench-Capon, M.J.R. Shave and B.M. Diaz, 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, 57-68 (1995).

24. F.P. Coenen, B. Beattie, B.M. Diaz, T.J.M. Bench-Capon and M.J.R. Shave, A temporal calculus for GIS using tesseral addressing, in M.A. Bramer and A.L. Macintosh (Eds), *Research and Development in Expert Systems XI, Proceedings of ES'94* 261-273 (1994).

25. F.P. Coenen, B. Beattie, B.M. Diaz, T.J.M. Bench-Capon and M.J.R. Shave, Temporal reasoning using tesseral addressing: towards an intelligent environmental impact assessment system, *Journal of Knowledge-Based Systems* **9**(5) 287-300 (1996).

26. A. Jeffrey, *Mathematics for engineers and scientists*, Van Nostrand Reinhold, Wokingham (1995).

27. G. Retz-Schmidt, Various views on spatial preposition. *AI Magazine* **9**(2) 95-105 (1988).

28. C.L. Hamblin, Instants and intervals, *Studium Generale* **24** 127-134 (1971).

29. J. Malik and T.O. Binford, Reasoning in time and space. *Proceedings, International Joint Conference on Artificial Intelligence (IJCAI'83)* **1** 343-345 (1983).

30. S.K. Chang, Q.Y. Shi and C.W. Yan, Iconic Indexing by 2-D strings, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **PAM1-9**(3) 413-428 (1987).

31. Z. Cui, A.G. Cohn and D.A. Randell, Qualitative simulation based on a logic of space and time, *AISB Workshop Proceedings on Qualitative and Causal Reasoning* (QCR'93), University of Birmingham (1983).

32. M.J. Egenhofer, Deriving the composition of binary topological relations, *Journal of Visual Languages and Computing* **5** 133-149 (1994).

33. A.G. Cohn, J.M. Gooday, B. Bennet and N.M. Gotts, A logical approach to representing and reasoning about space, *Artificial Intelligence Review* **9** 255-259 (1995).

34. D. Hernández, Relative representation of spatial knowledge: the 2-D case, in D.M. Mark and A.U. Frank, *Cognitive and Linguistic Aspects of Geographic Space*, Kluwer, Dordrecht, Netherlands, 373-385 (1991).

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

36. A.K. Mackworth, Consistency in networks of relations *AI Journal* **8**(1) 99-118 (1977).

37. J-L. Lauriere, A language and a program for stating and solving combinatorial problems, *Artificial Intelligence* **10**(1) 29-127 (1978).

38. M. Dincbas, P. van Hentenryck, H. Simonis, A. Aggoun, T. Graf and F. Berthier, The constraint logic programming language CHIP, *Proceedings of the International Conference on Fifth Generation Computer Systems (GGCS'88)*, Tokyo, Japan (1988).

39. A. Colmerauer, Opening the PROLOG-III universe, *Byte Magazine* **12**(9) (1987).

40. J. Jaffar and S. Michaylov, Methodology and implementation of a CLP system, *Fourth International Conference on Logic Programming*, Melbourne, Australia (1987).

41. J. Jaffar, J-L. Lasses and M.J. Maher, A logic programming language scheme, in D. DeGroot and G. Linderstrom (Eds), *Logic Programming, Relations, Functions and equations*, Prentice-hall, Englewood Cliffs, NJ, USA (1986).

Figure 1: *One-dimensional object space*

Figure 2: *One-dimensional shape fitting scenario*

Figure 3: *Two-dimensional object space*

Figure 4: *Two-dimensional shape fitting scenario*

Figure 5: *Three-dimensional object space*

Figure 6: *Three-dimensional shape fitting scenario*

Figure 7: *Four-dimensional object space*

Figure 8: *Four-dimensional shape fitting scenario*

-- --

Bibliographical Sketches

Frans Coenen is a lecturer within the Department of Computer Science at the University of Liverpool. His current research interests include; spatio-temporal reasoning - particularly its implementation using tesseral approaches and its application to Geographic Information Systems (GIS), Knowledge Based Systems (KBS) and their validation, verification and maintenance, and the application of all forms of computer science to the maritime industry. He has published widely on all these subjects.

Bridget Beattie is currently employed by the Connect Centre of the Department of Computer Science, University of Liverpool, to teach on short courses in Internet education. She gained a BSc in Chemistry from the University of Sussex in 1967, a PGCE from the University of Manchester in 1986 and an MSc in Software Engineering at the University of Liverpool, in 1989. Her research interests include programming MIMD parallel architectures and spatial reasoning, as well as issues of gender in Computer Science.

Trevor Bench-Capon is a senior lecturer in the Department of Computer Science at the University of Liverpool. His current research interests include; legal information systems; the role of ontologies in knowledge based systems and the verification and validation knowledge based systems. He has published three books and many papers on these and other subjects.

Bernard Diaz is Senior Lecturer in spatial and temporal systems in the Department of Computer Science, the University of Liverpool. His research interests and recent publications cover tesseral methods, vision systems, and history data visualisation in distributed environments. He is a member of the KRAFT team researching networked knowledge sharing.

Michael Shave is Professor of Computer Science at the University of Liverpool. His research has centred on databases, knowledge analysis and knowledge representation. He is currently leading a team at Liverpool in the KRAFT project, a multi-site research program which is investigating knowledge fusion from distributed information systems. He is also interested in spatial reasoning for geographic information systems.