A Temporal Calculus for GIS using Tesseral Addressing

Frans Coenen, Bridget Beattie, Bernard Diaz, Trevor Bench-Capon and Michael Shave

Department of Computer Science, The University of Liverpool,

Chadwick Building, P.O. Box 147, Liverpool L69 3BX.

email: frans@uk.ac.liverpool.compsci.

ABSTRACT

Many temporal reasoning techniques have been proposed. These tend to be based on some concept of first order predicate logic. The techniques can easily be integrated into standard logic based systems such as Knowledge Based Systems (KBS). However, temporal reasoning techniques cannot be so easily integrated with systems that are not based on such a logic representation. One area where this is evident is in Geographic Information Systems (GIS) based on raster representations. In this paper we describe a temporal reasoning mechanism *the event space calculus* based on a tesseral representation of time. This representation maps onto a raster grid. A a result the calculus is precisely compatible with raster based GIS and thus supports temporal reasoning concerning the nature of GIS objects while still permitting standard spatial operations. The calculus also supports both the notion of instantaneous and non-instantaneous events and intervals between events. In addition it permits all the standard operations found in more traditional temporal reasoning systems.

KEY WORDS: Tesseral temporal reasoning, Temporal reasoning, the Event Space Calculus, GIS, dGKBIS, TESTER.

1. INTRODUCTION

Many temporal reasoning techniques have been proposed, examples include the Situation Calculus (McCarthy and Hayes 1969), the Event Calculus (Kowalski and Sergot 1986) and the Interval Calculus (Allen 1983). These calculi (and others) are concerned with the temporal aspects of events distributed through time and tend to be formulated using first order logic. Such calculi have been integrated into many applications; examples include event scheduling in steel making plants (Dorn and Shams 1991) and robot arm control (Lin and Ionescu 1991). However, this integration requires a uniform logic representation which is not suited to all applications. The dynamic Geographic Knowledge Based Information Systems (dGKBIS) project currently in progress within the department of Computer Science at the University of Liverpool is investigating the integration of Geographic Information System (GIS) and KBS techniques. One aspect of this, the "dynamic" aspect, is the integration of temporal logic with GIS where the individual representations used are not uniform.

One GIS representation which is currently increasing in popularity is based on a raster grid which in turn maps onto a tesseral representation (see Samet 1990 or Diaz and Bell 1987). In this representation two and three dimensional space is continuously subdivided until a desired resolution (grain size or atomic level) is reached. Thus, given that temporal reasoning systems are based on some notion of formal logic and that we require a uniform representation to facilitate the desired integration, we can either use a logic representation for our GIS or a tesseral representation for our temporal logic. This paper is concerned with the latter option - a temporal logic based on a tesseral representation.

To manipulate temporal data using a tesseral representation we consider time to be a spatial dimension and have developed the concept of an *event space* in which temporally distributed events are contained. The principal advantage offered is that the event space can be hierarchically tessellated (Bell et al. 1983) to achieve compatibility with GIS representations. To support our notion of an event space we have also developed an *event space calculus*. We use the term calculus to relate it to the many other temporal calculi that have been proposed. The event space calculus is non-monotonic in that data in the event space can be amended or added to with the consequence that assumptions concerning existing events in the event space may be updated and changed. We have not (in common with many similar proposals) included a delete capability because this goes against the concept of non-destructive assignment espoused by classical logic. In line with other researchers (for example Jixin and Knight 1994) we also believe that any truly useful temporal reasoning system must be able to handle both events and intervals. Thus the event space calculus has been designed to operate using both types of temporal entity. We define events as temporal entities which have no duration (i.e. they are instantaneous). Intervals are then defined as events which have some duration associated with them marked by (instantaneous) start and end events. We have formalised the event space calculus using the Horn clause subset of classical logic (with negation as failure). The result is an executable logic which we have implemented in the form of a PROLOG module which in turn acts as a shell on top of which application specific code can be written. We refer to this shell as the TESTER (TESseral TEmporal Reasoning) Shell.

In the following sections both the event space concept and the supporting event space calculus are described in detail. The result is a novel approach to temporal reasoning which has the significant advantage that it is immediately compatible with tesseral raster GIS representations of space. It is also clear that such a representation is not incompatible with standard layered vector GIS, however implementation becomes a key issue, and as identified by Langran (1993), may become an over-riding concern.

2. LABELLING

Temporal reasoning techniques use some form of labelling convention to label events and intervals. These labelling conventions can be either implicit or explicit. In the first case events are represented in the form of a graph with the relationship between events implied through the edges connecting the nodes in the network; temporal reasoning about the events displayed then decomposes into a path-finding exercise. In the second case, events are tagged with a temporal label to which some form of first order logic can be applied to manipulate and reason about the tagged events. The simplest mechanism for tagging events is to assign a sequential number to events in the order in which they occur. Alternatively we can *time stamp* events. The advantage offered here is that they are guaranteed to be linear, thus we can determine intervals between events. Time stamping representations also support the notion of a *dense* representation where a third event can always be inserted between any two other events. The disadvantage of time stamping is that it is only applicable where an event has a known time associated with it. This restricts the scope of the reasoning that can be implemented. Most temporal reasoning systems thus also support tagging using *descriptors* of the form *before*, *after*, *during*, *overlapping* and so on. With respect to intervals Allen (1983) has identified thirteen such labels which represent the complete set of relationships between intervals.

The significant difference between the event space calculus described here and other temporal calculi is that in the event space calculus events and intervals are labelled using tesseral addresses. This mechanism has the advantage that it maps precisely onto raster representations of spatial data, allowing uniform treatment of time as a third or fourth dimension. The representation also supports labelling using relational descriptors , defined in tesseral terms, and all the standard temporal reasoning operations found in other temporal calculi.

3. THE TESSERAL LABELLING MECHANISM

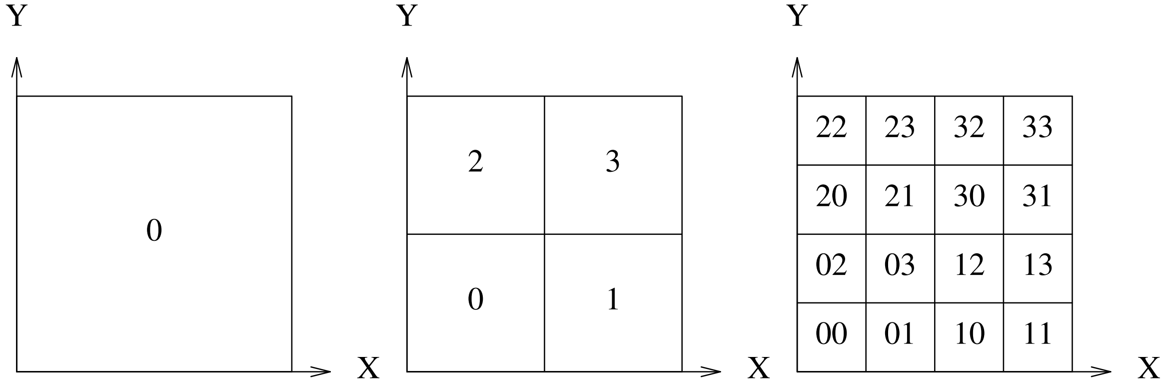
There is not sufficient space in this paper to give a full discussion of tesseral addressing, but the following points need to be mentioned:

1. Tesseral addressing is usually applied to two dimensional space but can equally well be extended to any number of dimensions (although there seems little point in going beyond four).

2. The principle comprises hierarchical amalgamation (Holroyd 1983) of a base atomic isohedral tiling (Grunbaum & Shephard 1977) to produce tesseral addresses for both the atomic and molecular tiles generated.

3. Tesseral addresses are constructed (*addressed*) in such a way as to reflect the tessellation process. There are many tesseral addressing conventions. One of the most straightforward is illustrated in Figures 1 (a), (b) and (c). Here, we commence with a (2-dimensional) space to which we assign the tesseral address 0 (Figure 1(a)). We then quarter it giving each quarter a unique number (Figure 1(b)). We can continue quartering the quadrants and assigning addresses as shown in Figure 1(c).

4. A true tesseral address is not limited to one quarter of the Euclidean plane. Using the concept of "beaked" digits (Diaz 1987) addressing can be extended over the entire Euclidean plane.



(a) Depth 0 (b) Depth 1 (c) Depth 2

Figure 1: *Tesseral addressing*

5. Using the numbering mechanism illustrated in Figure 1 we can see that the number of tesseral digits used reflects the hierarchical depth and thus the size of the tile addressed. If we have chosen the atomic level incorrectly and need a finer resolution we have two options. We can use fractional tesseral addresses (as described in Diaz 1987) or we can recast the tiling one level down. However, the latter option is only possible if our base tiling is square, or one of the three triangular isohedral tilings (Bell et al. 1983). (This is because only these four tilings are infinitely decomposable).

6. The tesseral addresses for two dimensional space can be constructed using two binary digits per tesseral digit, and by extension three binary digits in 3D space.

7. Having established a set of tesseral addresses associated with a particular space, only the addresses of interest need be stored in the database together with any associated *attribute* information. This has the advantage that all queries can be addressed directly to the database and the original spatial representation can be dispensed with.

8. Tesseral addressing is more than simply a numbering convention; it is also supported by an arithmetic and algebra. Tesseral arithmetic has parallels with conventional arithmetic, however, operations such as addition and subtraction imply translation (multiplication and division implies rotation and scaling). For example, with respect to Figure 1:

0+1=1, 1+1=10, 10+1=11 *etc.*

allows us to moves along the X-axis, while:

0+3=3, 3+3=30, 30+3=33 *etc.*

allows us to moves along the diagonal.

4. THE EVENT SPACE

To permit event and interval labelling using a tesseral addressing system we must consider events and intervals to be spatial entities, i.e. they must have some dimension associated with them. This idea in itself is not new and has a well documented history (see Gail Langran (1993) for a critical review and a summary), however, the application of the idea to the concept of an *event space*, in which temporally displaced events and intervals are contained, represents a novel departure from previous work. The event space in its current form is bounded by an application dependent start and an end tesseral address. Events and intervals within the event space are then referenced using any of the intervening available tesseral addresses. This convention is entirely consistent with standard tesseral addressing practice and is also consistent with related work in temporal reasoning - for example Kowalski's event calculus (Kowalski and Sergot 1986)

Thus events are represented by a single tessellation. However, whatever resolution (atomic tiling/grain size) is selected, tesserals will always have some dimension associated with them. If we consider the minimum tessellation shown in Figure 2(a) where the X-axis represents time, the tessellation has a duration of `n`. The effect of this is that (instantaneous) events appear to have some duration associated with them. To deal with this problem we have adopted the convention that events are represented by the leading edges of the tessellations. Intervals are then represented in the event space by a start and end event (Figure 2(b)). The duration of the interval given in Figure 2(b) is then equivalent to the tesseral length $|n-m|$. This convention is entirely consistent with standard tesseral addressing practice and is also consistent with related work in temporal reasoning - for example Kowalski's event calculus (Kowalski and Sergot 1986)

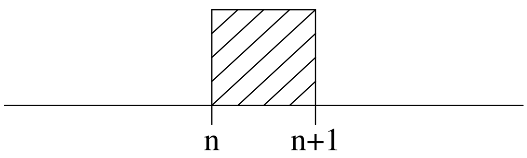


Figure 2(a): *Tesseral Event*

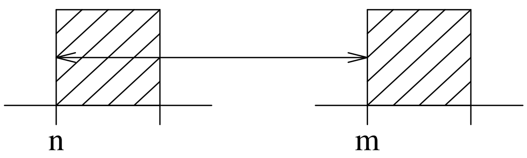


Figure 2(b): *Tesseral Interval*

Given that the event space must currently have some minimum grain size associated with it a further problem concerning the density of the representation must be considered. A dense representation requires that a third event can always be introduced between two other events, and that a further interval can always be placed within any other interval. Using the event space concept as outlined above the representation cannot be said to be a dense representation. It is not possible to add a third event between two other events if those two events are separated by only a single tessellation. Similarly it is not possible to place an interval within an existing interval if the duration of that interval is equivalent to a single atomic tesselation. Work is currently in progress to look at these problems by adopting more sophisticated tesseral addressing mechanism along the lines suggested in point 5 above. It should also be appreciated here that any number of unrelated events can share the same tesseral address; the event space calculus does not imply any specific relationship between such events.

Events in the event space can be categorised as being either free or referenced, the difference is that a free event has no fixed tesseral address associated with it while a referenced event has such an address. Events can also be related to one another using one of the following relationships, `before`, `after`, `equal`, `before-fixed` and `after-fixed`. These are defined as shown in Table 1. Events can also be related to intervals using the relationships given in Table 2.

```
holds E1 before E2        holds E1 after E2         holds E1 equal E2
        if TE1 < TE2              if TE1 > TE2              if TE1 = TE2

holds E1 before_fixed E2 by Td          holds E1 after_fixed E2 by Td
        if TE1 = TE2-Td                         if TE1 = TE2+Td

where:
        E1 and E2   = Event identifiers
        TE1         = Tesseral address associated with event E1
        TE2         = Tesseral address associated with event E2
        Td          = A tesseral duration
```

Table 1: *event - event relationships*

In the event space intervals are defined in terms of their start and end events. Thus an interval `I` might have a start event `Is` and an end event `Ie` associated with it which will have the tesseral addresses `TIs` and `TIe` respectively. These events will then be related using the event label `after` given in Table 1. Thus:

```
holds E before I         holds E during I         holds E starts I
        if TE < TIs              if TE > TIs              if TE = TIs
                                 and TE < TIe

holds E after I                                   holds E finishes I
        if TE > TIe                                       if TE = TIe

holds E before_fixed I by Td              holds E after_fixed I by Td
        if TE = TIs-Td                         if TE1 = TIe+Td

where:
        E   = An event identifier
        I   = An interval identifier
        TE  = Tesseral address associated with an event E
        TIs = Tesseral address associated with start of interval I
        Tle = Tesseral address associated with end of interval I
        Td  = A tesseral duration
```

Table 2: *event - interval relationships*

```
Ee after Es
```

Such an interval is considered to be a *variable* interval because it has no fixed duration. An interval that does have a fixed duration is referred to as a *constant* interval and is defined using the event label `after_fixed`. For example an interval `I` with a duration of `Td` would be defined as:

```
Ee after-fixed Es by Td
```

Thus intervals can be either variable or constant. Intervals can also be free or referenced. This gives rise to five possible combinations:

1. Free variable intervals

   Intervals which have no fixed start or end, or fixed duration.

2. Start referenced variable intervals

   Intervals which have a fixed start but an unknown duration.

3. End referenced variable intervals

   Intervals which have a fixed end but an unknown duration.

4. Free constant intervals

   Intervals with a fixed duration but no referenced start or end.

5. Referenced constant intervals

   Intervals with a fixed duration and a referenced start and end.

Intervals can also be related to other intervals or events. Relationships between intervals can be expressed using one or more of Allen's thirteen interval relationships (Allen 1983) augmented by by the two quantified labels - `before_fixed` and `after_fixed`. Similarly relationships between intervals and events can be expressed using the relationships `before`, `after`, `started-by`, `finished-by`, `contains`, `before_fixed` and `after_fixed`. The available relationships between intervals are defined in Table 3 and the relationships between intervals and events in Table 4.

Inspection of the labels given in Tables 1, 2, 3 and 4 indicates that all of the 34 relationships which they describe are defined in terms of the five event labels given in table 1. Hence we refer to these five labels as *primitive labels*. If desired further application dependent relationships can be defined on top of the TESTER shell using any of the available relationship definitions.

5. THE EVENTBASE

In a tesseral GIS, information concerning spatial entities can be stored in a database with a tesseral address acting as the key to specific rows in tables. We have adopted a similar approach with respect to our event space and have developed what we term an *eventbase*. Currently the eventbase contains three tables, an event table, a relationship table and an interval table. In the event table information concerning events in the event space is stored. The table has three fields:

1. A tesseral address field (Ta) describing the event's location in the event space.
2. An event identifier field (E) in which an event identifier, comprising a single string supplied by the user, can be placed.
3. An event type label field indicating whether the tesseral address is fixed (referenced) or assumed (free).

Data concerning the relationship between events is stored in the relationship table. The table has four fields:

1. An event identifier field (E1).
2. A relationship field in which an appropriate relational label can be placed.
3. A quantifier field which can be used to include information to quantify the label in a relationship field.

```
holds I1 before I2      holds I1 during I2      holds I1 overlaps I2
        if TI1e < TI2s          if TI1s > TI2s          if TI1s > TI2s
        or TI1e = TI2s          and TI1e < TI2e         and TI1s < TI2e
                                                        and TI1e > TI2e

holds I1 after I2       holds I1 contains I2    holds I1 overlapped-by I2
        if TI1s > TI2e          if TI1s < TI2s          if TI1s < TI2s
        or TI1s = TI2e          and TI1e > TI2e         and TI1e > TI2s
                                                        and TI1e < TI2e

holds I1 meets I2       holds I1 starts I2      holds I1 finishes I2
        if TI1e = TI2s          if TI1s = TI2s          if TI1e = TI2e
                                and TI1e < TI2e         and TI1s > TI2s

holds I1 met-by I2      holds I1 started-by I2  holds I1 finished-by I2
        if TI1s = TI2e          if TI1s = TI2s          if TI1e = TI2e
                                and TI1e > TI2e         and TI1s < Ti2s
                        holds I1 equals I2
                                if TI1s = TI2s
                                and TI1e = TI2e

holds I1 before_fixed I2 by Td              holds I1 after_fixed I2 by Td
        if TI1e = TI2s-Td                           if TI1s = TI2e+Td

where:
        I1 and I2 = Interval identifiers
        TI1s      = Tesseral address associated with start of interval I1
        TI1e      = Tesseral address associated with end of interval I1
        TI2s      = Tesseral address associated with start of interval I2
        TI2e      = Tesseral address associated with end of interval I2
        Td        = A tesseral duration
```

Table 3: *interval - interval relationships*

4. A second event identifier field (E2) for the event identifier associated with the event to which the first event is related.

The interval table is then used to store information concerning events. The table has three fields.

Interval Table

1. An interval identifier (I) field in which an interval identifier, comprising a single string supplied by the user, can be placed.

```
holds I before E         holds I contains E      holds I started-by
        if TIe < TE              if TIs < TE             if TIs = TE
                                 and TIe > TE
holds I after E                                  holds I finished-by E
        if TIs > TE                                      if T1e = TE

holds I before_fixed E by Td               holds I after_fixed E by Td
        if TI1e = TE-Td                            if TI1s = TE+Td
where:
        I   = An interval identifiers
        E   = An event identifiers
        TIs = Tesseral address associated with start of interval I
        TIe = Tesseral address associated with end of interval I
        TE  = Tesseral address associated with event E
        Td        = A tesseral duration
```

Table 4: *interval - event relationships*

2. A start event identifier field (Es) for an identifier generated by the system by appending the string `_start` to the interval identifier.
3. An end event identifier field (Ee) for an identifier generated by the system by appending the string `_end` to the interval identifier.

Each start and end identifier associated with a particular interval identifier in the interval table will also appear in the event table. They will also appear in the relational table such that the end event is defined as occurring after the start event. Thus the event identifiers are the key between the three tables.

Queries can be directed at the eventbase in the usual manner. A number of styles of query are supported. Firstly we can request information concerning specific events or intervals, or all events and/or intervals. Secondly we can request information concerning a specific time, i.e. *What events and intervals occur at time T*. We can also express queries concerning events and intervals occurring before or after a specific time, or over a range of times. Lastly, using the relational labels given in Tables 1, 2, 3 and 4 we can formulate relational queries such as *Does interval I1 occur during interval I2?*, or *What events and intervals occur during interval I2?*, or *To what intervals is interval I2 related by the relationship during?* or *What intervals and events occur during other events?*. It should also be noted that responses to queries indicate whether events or intervals have assumed or referenced addresses associated with them.

6. THE EVENT SPACE CALCULUS

We have seen that events and intervals in the event space can be free or referenced. We have also noted that if we wish to treat time as a spatial dimension we must attribute boundary dimensions to events and intervals represented in the event space. This is a trivial operation in the case of fixed events and intervals: we simply translate the given time(s) into a tesseral dimension and add the event or interval to the eventbase. In the case of free events, or events referenced only through some relationship to another event or interval (which might itself be unreferenced) an assumed tesseral address must be generated. This is implemented using the event space calculus. Where unreferenced events and intervals are added to the event space the calculus will assign assumed tesseral addresses to them. Further, as each new event or interval is added (or updated), the calculus will reassess all previous assumptions concerning existing events and intervals.

When adding or updating events the calculus supports the following operations:

1. Add a free event.
2. Add a referenced event.
3. Update a free event by adding a reference.
4. Add an event related to some existing event or interval using a label.
5. Update an existing event by relating it to some other existing event or interval using a label.

Similarly when adding or updating intervals the event space calculus supports the following operations:

1. Add a free constant or variable interval.
2. Add a referenced constant or variable interval.
3. Update an interval by adding a reference.
4. Update a variable interval by adding a duration.
5. Add a constant or variable interval related to some existing event or interval using a label.
6. Update an existing interval by relating it to some other existing event or interval using a label.

Where a free event is added to the eventbase the event is assumed to have a tesseral address of `0`. In the case of free variable intervals the start event is assigned a tesseral address of `0` and the end event a tesseral address of `0+K` where `K` is equivalent to 1, 2, 4 or 8 according to whether we are operating in one, two, three or four dimensions respectively. In the case of a free constant interval the start and end tesseral addresses will be `0` and `0+Td`, where `Td` is the tesseral equivalent of the intervals duration.

Where, in the absence of any other information, an event (or interval) E1 can only be assigned an assumed tesseral address because of some relationship to another event (or interval) E2 the address is assumed by reference to one or more of the five primitive relational labels on which all the 34 relationships supported by the event space calculus are based. In some cases the resulting assumed tesseral address may be less than 0 or greater than the maximum tesseral address permitted by the selected tesseral resolution for the event space. In this case, if E2 also has an assumed tesseral address associated with it the event calculus will attempt to revise this assumption. If this is not possible, or E2 is a referenced event, the initial event space addition or update will fail. This is a direct consequence of the non-dense representation currently used, however, it is expected that this problem will be resolved with the adoption of a more sophisticated tesseral addressing technique.

As events and intervals continue to be added or updated the event calculus revises its assumptions according to the available information concerning existing events and intervals. For each assumption there are three possibilities.

1. The assumption holds.
2. The assumption does not hold thus the calculus will attempt to adjust the assumption.
3. Failing 2 the calculus will attempt to adjust the tesseral address of the event on which the current assumption is dependent.

Where an assumption cannot be updated because of the particular configuration of the events and intervals represented in the event space the current addition or update will fail.

7. OPERATION

To illustrate the power of the event space calculus described here it should be fully integrated with a three dimensional spatial reasoning mechanism so that four dimensional reasoning can take place. However, in the absence of such a mechanism, we can give a flavour of the operation of the event calculus by considering only the temporal dimension (i.e one dimensional space). As a result the constant `K` (see above) will be equivalent to the value `1`.

Consider the following facts concerning a number of temporally displaced events and intervals occurring at some point in a four dimensional space:

```
1) a is a free event
2) b is a variable interval finished-by the event a
3) c is a variable interval overlapped by the interval c
4) d is a constant interval with a duration of 2 tesserals
   which meets interval b
5) e is an event referenced to tesseral 5 which occurs
   during interval b
6) f is a interval that contains interval b
7) g is a constant interval with a duration of 2 tesserals
   which is started by event e
```

We will add each of these facts to the event space in turn and consider the assumptions that are made.

Initially event `a`, when entered into the event space, is assumed to occur at a tesseral address of `0`. When interval `b` is added this assumption is revised. The interval is assumed to have a duration of 1 (remember that this example is only operating in one dimensional space) hence its start will be allocated the tesseral address `0` and its end the tesseral address `1`. The revised address of event `a` is now `1` by virtue of the definition of the relationship `finished-by` (see Table 4). The eventbase will now contain information concerning the event `a` and the interval `b` as shown in Table 5, and the event space will be of the form shown in Figure 3(a).

Event Table			Relationship Table				Interval Table		
E	Ta	Type	E1	Label	Q	E2	I	Es	Ee
a b_start b_end	1 0 1	free free free	a b_end b_end	equal after equal	- - -	b_end b_start a	b	b_start	b_end

Table 5: *Eventbase (1)*

The definition for the label `overlapped-by` states that the start of the first interval must occur within the second interval, and that the end of the first interval must occur after the end of the second interval (see Table 3). Thus when we add the interval `c` to the event space the duration of interval `b` will be extended to accommodate the start of `c`. The event space will now be of the form shown in Figure 3(b).

To add a constant interval related to some other existing interval or event this must be implemented in two steps. First we must create the interval and then add the relationship. Similarly with respect to referenced events related to some existing interval or event we must first create the event and then add the relationship. We must therefore first break down facts 4 and 5 into their constituent parts before we can add them to the eventbase. Thus:

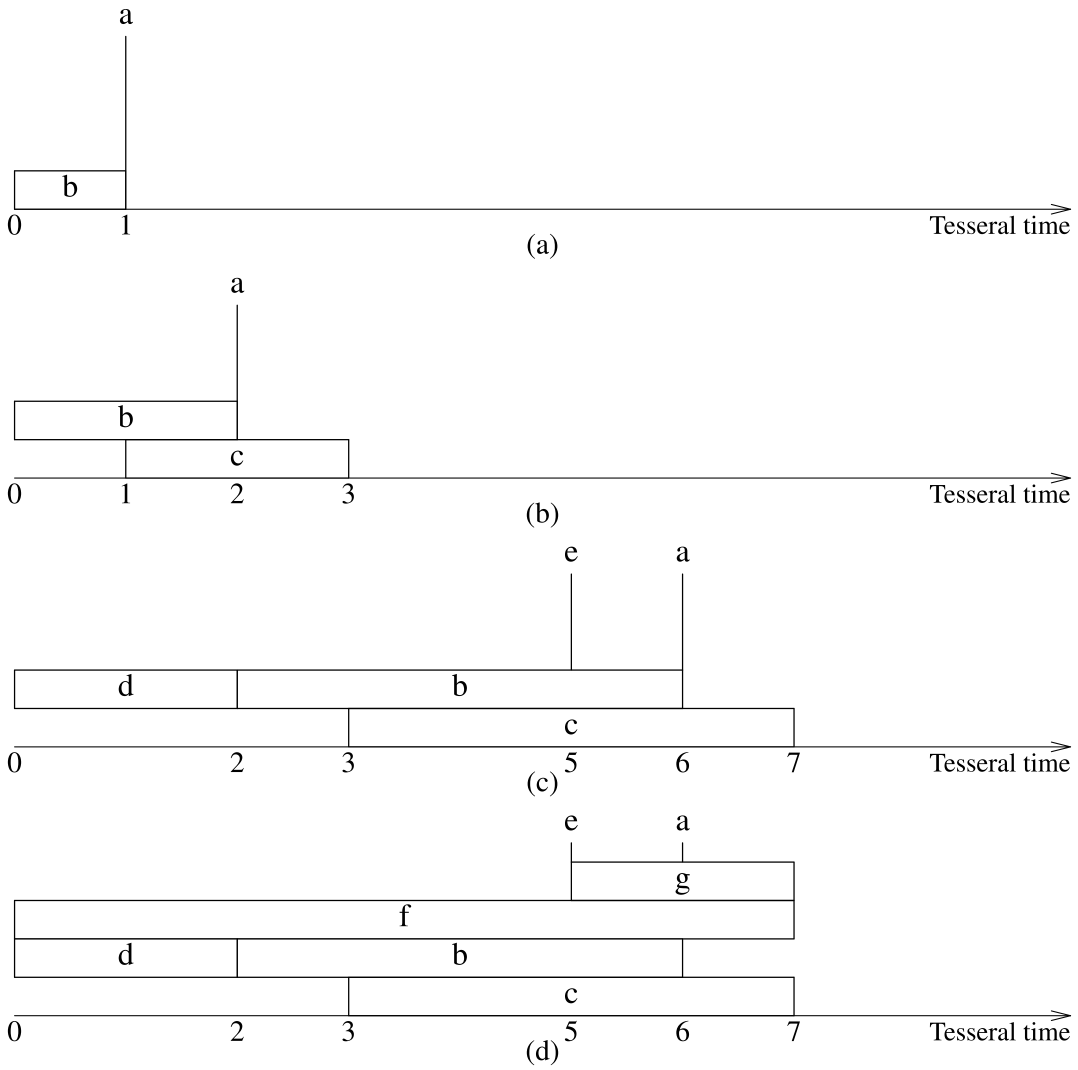


Figure 3: *Event space calculus operation*

```
d is a constant interval with a duration of 2 tesserals
d is an interval that meets interval b
e is an event referenced to tesseral 5
e is an event that occurs during interval b
```

Alternatively we could write a "macro" to sit on top of the TESTER shell to perform the same task. Whatever the case, as a result of adding interval `d` the assumptions for `a`, `b` and `c` will all be updated by (effectively) adding a tesseral 2 to each address. If we then also add the event `e` referenced to the tesseral address 5 this will have the effect of updating the assumed tesseral address of the end event associated with the interval `b` in accordance with the `during` relationship given in Table 2. This in turn will cause the assumptions associated with the event marking the end of interval `c` (which must be after the end of interval `b`) to be revised. The event space will now be of the form shown in Figure 3(c).

When we add interval `f` it is allocated an assumed start address of `0` and an assumed end address occurring after interval `b` (`f` is defined as "containing" `b`) of `6`. In the case of event `g` this must again be broken down into its constituent facts:

```
g is a constant interval with a duration of 2 tesserals
g is an interval that is started by event e
```

before it can be added to the event space. As a result `g` will be given a start address of `5` (equal to event `e`) and an end address of `7`. In this case the start of `g` will be referenced because event `e` is a referenced event. Similarly the end of `g` will also be referenced because it occurs at a fixed time (2 tesserals) after the start (which is now referenced). The inclusion of intervals `f` and `g` will not effect any of the assumptions associated with existing events and entities. The event space will now be of the form given in Figure 3(d) and the eventbase of the form given in Table 6.

Event Table			Relationship Table				Interval Table		
E	Ta	Type	E1	Label	Q	E2	I	Es	Ee
d_start	0	free	c_start	before	–	b_end	d	d_start	d_end
c_start	3	free	c_start	after	–	b_start	c	c_start	c_end
d_end	2	free	d_end	equal	–	b_start			
			d_end	after_fixed	4	d_start			
c_end	7	free	c_end	after	–	b_end			
			c_end	after	–	c_start			
a	6	free	a	equal	–	b_end			
f_start	0	free	f_start	before	–	b_start	f	f_start	f_end
b_start	4	free	b_start	after	–	f_start	b	b_start	b_end
			b_start	before	–	e			
			b_start	equal	–	d_end			
			b_start	before	–	c_start			
f_end	7	free	f_end	after	–	b_end			
			f_end	after	–	f_start			
b_end	6	free	b_end	before	–	f_end			
			b_end	after	–	e			
			b_end	after	–	c_start			
			b_end	before	–	c_end			
			b_end	equal	–	a			
			b_end	after	–	b_start			
e	5	ref	e	equal	–	g_start			
			e	before	–	b_end			
			e	after	–	b_start			
g_end	7	ref	g_end	after_fixed	4	g_start			
g_start	5	ref	g_start	equal	–	e	g	g_start	g_end

Table 6: *Eventbase (2)*

In this manner the user can continue to update and add to the eventbase/event space. At any time during the process the eventbase can of course be queried and information returned.

8. CONCLUSIONS

A temporal reasoning system based on a tesseral labelling technique has been described. The system comprises the event space calculus which manipulates a conceptual event space and an eventbase which is used to store this space. The advantage offered is that the representation is precisely compatible with that used by tesseral GIS, hence facilitating temporal reasoning concerning GIS objects.

Although the ramifications of using tesserals are still being explored, we consider that the reasoning technique described has provided a firm base on which further investigation concerning the integration of KBS and GIS can be built. Current work is focussed on two areas. A more sophisticated tesseral temporal representation to provide support for dense temporal reasoning; and the integration of temporal and spatial reasoning within an unified tesseral representation. It is envisaged that the latter, when used with the tesseral temporal reasoning system described here, will result in a powerful spatio-temporal reasoning system which will be generally applicable to all types of GIS.

REFERENCES

1. Allen, J.F. (1983). Maintaining Knowledge about Temporal Intervals. Communications of the ACM, Vol 26, No 11, pp832-843.
2. Bell, S.B.M., Diaz, S.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.
3. Diaz, B.M. (1987) Tabular Tesseral Methods and Calculators. In Diaz, B.M. and Bell, S.B.M. (eds), Spatial Data Processing using Tesseral Methods, Natural Environment Research Council publication, Swindon, England, pp832-843.
4. Diaz, B.M. and Bell, S.B.M. (1987). Spatial Data Processing using Tesseral Methods. Natural Environment Research Council publication, Swindon, England.
5. Dorn, J. and Shams, R. (1991). An Expert System for Scheduling in a Steel Making Plant. Proceedings World Congress on Expert Systems (WCES'91), Pergamon Press, New York, pp395-404.
6. Grunbaum, B. and Shephard, G.C. (1977). The Eighty-one Types of Isohedral Tilings in the Plane. Procs. Cambridge Philos. Soc., Vol 82, pp177-196.
7. Holroyd, F.C. (1983). The Geometry of Tiling Hierarchies. Ars Combinatoria, Vol 16B, pp211-244.

8. Jixin, M. and Knight, B. (1994). A General Temporal Theory. The Computer Journal, Vol 37, No 2, pp114-123.

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

10. Langran, G. (1993). Time in Geographic Information Systems. Taylor & Francis Ltd, London.

11. Lin, Jing-Yue and Ionescu, D. (1991). A Controller Approach for Reasoning in a Temporal Logic Framework. Proceedings World Congress on Expert Systems (WCES'91), Pergamon Press, New York, pp1609-1616.

12. McCarthy, J. and Hayes, P.J. (1969). Some philosophical Problems from the Standpoint of Artificial Intelligence. In Meltzer, B. and Michie, D. (eds), Machine Intelligence 4, Edinburgh University Press, Edinburgh.

13. Samet, H. (1990). Applications of Spatial Data Structures, Addison-Wesley, New York.