HIERARCHICAL SPATIAL REASONING

Andrew U. Frank
Dept. of Geoinformation
Technical University Vienna
frank@geoinfo.tuwien.ac.at

Abstract

Using hierarchical methods for spatial reasoning is a popular research topic. Hierarchical spatial data structures, especially quadtrees, are used in many implementations of GIS and have proved their efficiency. Operations on hierarchical spatial data structures are effective to compute spatial relations, but do not automatically imply Hierarchical Spatial Reasoning. Hierarchical spatial reasoning is using coarser, less detailed representations to compute an *approximative answer* if the quality of the approximation is sufficient. Hierarchical spatial reasoning is closely related to computing approximative results and estimation of their errors.

This paper explores two spatial reasoning operations and deduces a general definition of 'hierarchical spatial reasoning'. Although the examples are very simple - computation of area and intersection - and applied to a raster representation, the definition appears general. Compared with other definitions it captures much of the essence of hierarchical spatial reasoning. This sets the framework in which general rules when hierarchical spatial reasoning can be employed may be deduced.

Hierarchical data structures are useful for hierarchical reasoning, but they can be transformed to a more efficient 'incremental hierarchical structure', e.g., an incremental quadtree. Then, incremental hierarchical spatial reasoning algorithms use previously computed values to compute the next approximation and are therefore as efficient as a direct calculation with the same error bound.

1 Introduction

Hierarchical data structures are very popular in computer science, because they support the 'divide-and-conquer' strategy for algorithms efficiently. The best known is the quadtree structure, with a large number of variants (Samet 1989a; Samet 1989b). There are also other efforts for hierarchical structures in spatial information theory, defining hierarchical divisions of curves (Ballard 1981), hierarchical triangulation (De Floriani and Puppo 1992; Puppo and Dettori 1995), and hierarchical networks (Car and Frank 1994b). Timpf has shown how a spatial task is broken down hierarchically into subtasks (Timpf et al. 1992) and Voisard has formalized similar task subdivisions for a mapping system (Rigaux et al. 1993; Voisard and Schweppe 1994).

In a recent discussion, the five characteristics of spatial knowledge, which make spatial reasoning difficult, were listed as:

- incomplete
- imprecise
- hierarchical

- qualitative
- time dependent or related to motion or process.

Progress in these areas is crucial for effective spatial reasoning systems. Recently, the term ‘hierarchical spatial reasoning’ has been used (Car and Frank 1994a; Jones and Luo 1994; Whigham 1993) and several examples explore such ideas (Abler et al., 1971; Dutton 1993; Fotheringam 1992; Goodchild and Shiren 1990; Hirtle and Jonides 1985; Noronha 1988). Golledge has included ‘hierarchical reasoning’ in a recent list of the most important open questions in spatial reasoning (Golledge 1992). Cohn has discussed coarser and finer representations (Cohn 1995), and Papadias and Glasgow (Papadias and Glasgow 1991) have extended symbolic projections to hierarchical situations. Different levels of spatial resolution are typical for topographic maps and map series of varying scale can be seen as hierarchies (Timpf and Frank 1995). Whigham extends the hierarchical structure to both space and time in which events happen (Whigham 1993). These efforts have provided some points in this wide and admittedly very important field of hierarchical reasoning, but a general framework for hierarchical spatial reasoning and a definition for it are still missing.

Hierarchical data structures have been developed for particular algorithms (Samet 1989a; Samet 1989b). It was found that the quadtree data structure with a nearly infinite number of variants is generally very useful for many different tasks and leads to efficient algorithms for spatial data with high levels of spatial autocorrelation.

Spatial reasoning is used here to denote any deduction of information from a representation of a spatial situation. It includes the simple calculation of spatial predicates, used here as running example (e.g. the area of a region), and spatial relations (i.e. the intersection of two areas), but also all operations of map algebra (Tomlin 1983a; Tomlin 1983b), symbolic spatial reasoning, etc. (Freksa 1991; Freksa 1992; Cohn - new submitted paper) and more complex tasks like planning of routes, allocation of resources etc.

Hierarchical spatial reasoning computes approximation with error bounds. This permits to decide when the result is ‘good enough’ for the task at hand and further, more detailed, computation is not warranted . Hierarchical spatial reasoning is the application of a non-hierarchical spatial reasoning algorithm to a series of increasingly finer (higher resolution) representations of the same situation, giving a series of improving approximations of reality. For each application of the algorithm not only the value is computed, but also bounds on the errors. These allow to determine when the approximation is good enough for the given task. A hierarchical operation is more efficient as it avoids exploring detailed levels if it can be decided on a coarser level that further detail does not contribute to improve the approximation.

Davis (1990, p. 270) points to advantages and disadvantages of spatial reasoning using occupancy arrays (as we do here) and his comments can be generalized. It is difficult

- to merge partial knowledge or knowledge at different levels of detail,
- to reason about cells which are only partially occupied.

The approach here - hierarchization of the knowledge base and to reason with error bounds - addresses both issues.

Content of the Article

In this paper I will explore hierarchical spatial reasoning based on a quadtree-like structure, assuming that quadtrees are a good, simple and well understood example used in spatial reasoning. Quadtrees are widely used due to their efficiency; they are also well explored theoretically (Samet 1989a; Samet 1989b). No particular use is made of the regularity of the tesselation and it is expected that the results hold for hierarchical spatial data structures which are not based on a regulartesselation (e.g., (De Floriani and Puppo 1992)).

Two spatial reasoning operations are used as examples: the computation of the area and the decision if two areas intersect. They are applied to a representation of regions as regular tessellation (raster). In this framework, the difference between algorithm on quadtrees and hierarchical spatial reasoning becomes clear and a formalization of the informal definition for hierarchical spatial reasoning given above emerges. The question remains what conditions a problem or an algorithm mustfulfil so that hierarchical processing is possible. Minimally, a method to assess the quality of the computed result is necessary, but possibly other conditions, e.g., order decomposability (Overmars 1983), must hold.

A special case can be identified: hierarchical spatial reasoning is more efficient if it is incremental, i.e. if the computation of each successive approximation uses the values computed for the previous approximation. A transformation of quadtrees to an 'incremental quadtree' is shown and demonstrated how it makes computation more efficient. The same 'breadth first' transformation can be applied to other hierarchical data structures. It remains to be seen for which class of algorithms 1) incremental computations are possible and when 2) error bounds can be computed for each incremental approximation. The framework developed here allows such an analysis.

The connection between hierarchical spatial reasoning and approximations and errors - another key research field today (Burrough and Frank 1996; Goodchild and Gopal 1989) - appears to be closer than expected. There are also connections to fractals, but these are not explored here.

This paper is structured as follows: the next section shows how common-sense spatial reasoning is widely used and is crucial for our understanding of space as a container. The following section introduces the two simple spatial reasoning tasks which are used as an example and gives the solution for a regular raster. The same tasks are then applied to quadtrees, which provides the contrasting case of a non-hierarchical algorithm applied to a hierarchical data structure. Section 5 transforms the same tasks to a hierarchical processing and generalizes to a definition of spatial reasoning. The simple case demonstrates the interaction between data structure and reasoning method clearly. It points to an improved incremental formulation. Incremental spatial reasoning with the corresponding incremental hierarchical data structures makes computation more efficient. Using a 'breadth first' transformation, incremental data structures can be constructed from regular ones. An incremental quadtree is given as an example. The paper concludes with a list of questions to investigate the limitations of this framework and which spatial reasoning tasks can be 'hierarchized'

2 Common Sense Hierarchical Spatial Reasoning

Humans reason about space and spatial properties, often using hierarchical reasoning structures. Observations of human reasoning suggest the two princips of hierarchical spatial reasoning as presented above:

- the need for abstraction to reduce the level of complexity, and
- the conceptualization of space as containers.

An additional principle, the hierarchical organisation of processes which are modelled to operate at different levels of resolution and interact in a hierarchical fashion, appears as a more complex extension of the methods presented here and should be treated in a separate paper.

2.1 Humans Use Hierarchical Reasoning in Spatial Situations

Humans apply hierarchical concepts to spatial situations very often. Space as a container (the in/out image schema of Lakoff (Lakoff and Johnson 1980)) is one of the most often used conceptual schemas. Containers lead to an inclusion relation, which is transitive and therefore gives rise to a hierarchy of inclusions. Human reasoning often uses this simple 'hierarchical transitivity':

- If an apple is in the fridge and the fridge is in the kitchen, then the apple is in the kitchen;

but also:

- if the apple is not in the kitchen, and the fridge is in the kitchen, then it is not in the fridge.

A very strong evidence of hierarchical effects in spatial reasoning is the observation that humans use this hierarchical reasoning based on spatial containers, even when not appropriate (Figure 1) (Stevens and Coupe 1978).

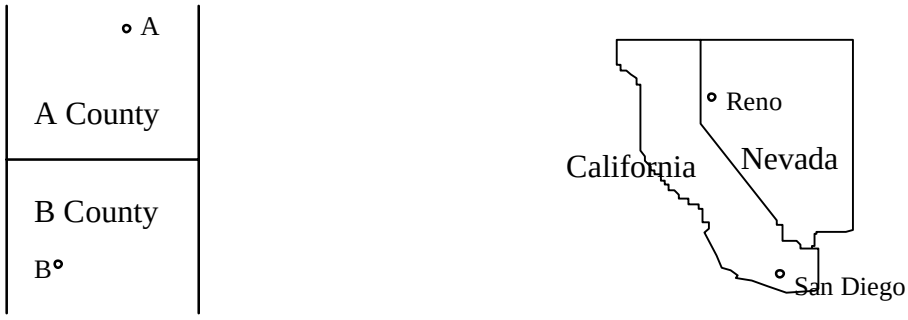

Figure 1 - For correct and error reasoning (according to (Stevens and Coupe 1978))

1) A in A County
2) B in B County
3) A County North of B County
-> A North of B

1) San Diego in California
2) Reno in Nevada
3) Nevada East of California
->?? Reno East of San Diego

2.2 Hierarchies Are Crucial for Spatial Reasoning

Space is continuous in two dimensions and can be observed from many different points of view and at different resolution. The same objects can be represented differently, depending on the purpose, indicating the point of view to take and the level of detail to be included. This is generally called 'multiple-representation' of spatial objects

(Buttenfield 1993; Buttenfield and Delotto 1989; NCGIA 1989; Timpf to appear). The representation of a spatial object at different levels of resolution leads to hierarchical representation: more and more details are included as one descends the hierarchy.

Spatial reasoning - here defined as any inference process with spatial properties - uses the level of detail appropriate for the task. This is an economic principle: a task is solved with the least amount of effort. Effort is reduced by selecting a level of detail which is sufficient for the task, but not finer. Working through more details does not contribute to the solution comparable with the cost of the additional computation: computing better solutions costs increasingly more, but contributes decreasingly to the solution.

This hierarchical reasoning works, because the interactions of processes are clustered around some typical level of detail: the size of everyday small objects and buildings are two typical levels of detail, where many interactions occur. For each task an appropriate level of detail is known from experience (Fraser 1981). Informally, one can observe many examples of this general behavior in everyday situations: while driving a car, one does not worry about its parts; while planning a building subdivision, the plants in the future gardens are not considered.

Brief, spatial reasoning is infinitely complex, because the spatial world is infinitely complex. Hierarchical structuring of this complexity is crucial to reduce the spatial reasoning tasks to manageable levels. Hierarchical spatial reasoning can be generalized and applied metaphorically to non-spatial situations, provided theyfulfil the necessary conditions that the rules are applicable.

3 Example Tasks and their Formalisation

Two simple tasks, one a spatial calculation, the other a test for a spatial relation, are applied to regions represented in a regular tessellation (raster). The tasks are well understood, and algorithms for the selected representations are well known. They lend themselves to a ‘natural’ form of hierarchical structuring, using quadtrees efficiently. These tasks are also simple enough to be formally described in the space allocated for a paper, here in the functional language Haskell (Hudak et al. 1992)(exactly the Gofer implementation of it (Jones 1991; Jones 1994)). The use of this functional language, which allows second order functions (i.e. functions which have functions as arguments), permits the separation of operations on the representation of spatial objects (resels) and on the data structure (quadtree). The two operations ‘intersect’ and ‘area’ are organised in two classes (Regions and Areas):

```
class Bools b => Regions a b where
     intersect ::  a -> a -> b
class (ZeroOne i, Num i) => Areas a i where
     area :: a -> i
```

3.1 Representation of Regions

A sharply delimited area is given as an array of resels (resolution elements (Tobler 1995)). The resels have two values: inside or outside the region, coded as I or O.

```
data Pixel2 = I2 | O2
```

3.2 Area as a Count of Black Resels

The standard method for computing the area in an array is to count the inside resels. A count operation working on a single resel (defined in the first instance) is sequentially applied to all resels in the array (in the second instance). This code is fully general and

will work for arbitrary data structures, provided the generalized folding operation `gfoldr` is defined.

```
instance Num1 i => Areas Pixel2 i where
     area I2 =  one1
     area O2 =  zero1
instance (Folds f, Areas a i) => Areas (f a) i where
     area a = gfoldr ((+).area) zero1 a
```

3.3 Intersection Operation

Two regions are intersecting if there is any resel inside both of them. For this we have to check for each corresponding pair of resels if we find a combination of two values, i.e. two resels in both (all other combinations yield `False`); mathematically, we observe that spatial 'intersect' and Boolean 'and' form the same algebra[1].

```
instance Regions Pixel2 Bool where
    intersect I2 I2 = True
    intersect _ _ = False
```

The test is then sequentially applied to all resels in the two arrays and the result combined with logical 'or' (for the test if 'any' resel is inside); this is coded generally for all data structures provided the logical folding operation 'any' (here `g2any`) is defined.

```
instance (Fold2s f, Regions a b) => Regions (f a) b where
     intersect a b = g2any intersect a b
footnote:
```

4 Quadtree Computation

Representing regions in an array fills much storage space and leads to slow processing, due to a large number of resels, which must be inspected. The effort for the computation is linear in the number of resels and therefore in the size of the array, independent of how much of the space is filled with objects of interest, how much is background etc.

Spatial data show much spatial correlation. An encoding where large areas with the same value are grouped together and encoded with a single value is therefore more efficient, for example run length encoding or quadtrees. It results in a much more compact representation and speeds up processing at the same time. The improvement is achieved with the small investment of initially building the data structure.

4.1 Definition of Quadtree

There are many different variants of quadtrees known (Samet 1989b), all based on the same principle of a 4-way branching tree data structure. Recursively, a quadtree is either a leaf (QV) or it is a tree with four quadtrees:

```
data Quad2 p = Q2 (Quad2 p)(Quad2 p)(Quad2 p)(Quad2 p) | QV2 p
```

4.2 Spatial Interpretation of Quadtrees

It is customary to interpret a quadtree structure as a representation of space, in which the leaf nodes are resels in a square array (Figure 2). Resels of higher level represent

[1] In functional languages the arguments to which a function is applied follow the function name and are not enclosed in parentheses, f(x) is written as f x. Parentheses are only used to group expressions. The two higher order functions gfoldr and g2any apply the function passed as a second argument to all elements in the data structure and combine the results. Areas are summed up with (+); (.) combines two functions, g2any tests if any value is true.
The code given here is simplified and leaves out some technical detail, running code can be obtained by ftp from the www homepage of the author (http://www.geoinfo.tuwien.ac.at.)

four times the area of the resel one level lower. The values of the resels are the color of the area, but other properties of the area can be represented as well.

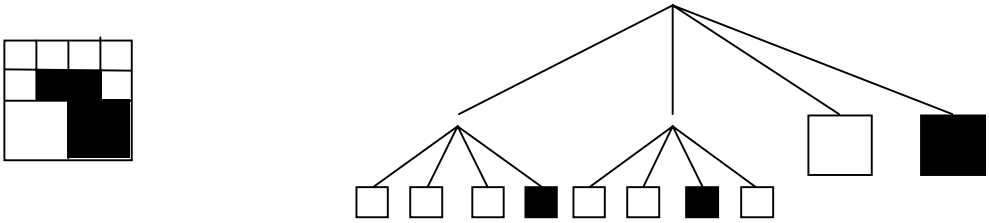

Figure 2 - Example of a quadtree

To keep the code simple, the size of the resel is included with each leaf of the quadtree; this can be left out in optimized code.

```
data QuadLeaf p  = QL Int p    -- the leaf size and the resel value
```

4.3 Area Count and Intersection Test in a Quadtree

Code to compute the area of such a leaf and the intersection of two leaves uses the corresponding operations for the resels and corrects for the size of the area.

```
instance (Num1 i, Areas p i) => Areas (QuadLeaf p) i where
    area (QL i p) = smult  (2^(i-1)) (area p)
instance (Regions p b, Num1 b) => Regions (QuadLeaf p) b where
    intersect (QL i p) (QL j q) = if (intersect i j) then …
```

These operations are 'spread' over the quadtree with the higher order functions `gfoldr` and `g2any` as given above. These functions require the definition of a `map` and a `fold` operation on the quadtree.

4.4 Quadtree as More Efficient Resel Arrays

Quadtrees are an effective method to represent areal data, and a number of specialized data structures and algorithms are known. For operations, quadtrees have the conceptual simplicity of arrays of resels, but allow more efficient processing. In general, the processing time of spatial algorithms which are based on comparison of resels (e.g. most operations from the map algebra (Tomlin 1983b; Tomlin 1989): intersection, union of areas, reclassification etc.), have a processing time, which is linear in the number of resels to be compared. As a quadtree has much less quad resels, processing is faster. The number of resels in a quadtree of fixed resolution is $O(l)$ (Samet 1989a, p. 6), where l is the length of the boundary of the region, and thus most algorithms are also linear in the length of the boundary (and not in the area of the regions).

4.5 Invariance of Operations Under the Transformation Between Array and Quadtree Representation

It is obvious that the result from the operations must be the same independent of the representation selected, thus the area calculation or the intersection test yield the same result, if applied to an array or a quadtree representation. The performance of the computation may be faster with the quadtree. It always computes a result which is completely accurate, and inspects therefore all details where there are such.

This invariance can be used to test the algorithms, but it is also possible to show in the abstract that the conversion from array to quadtree (or the reverse) does not affect the result. In terms of category theory, the following diagram commutes (Figure3):

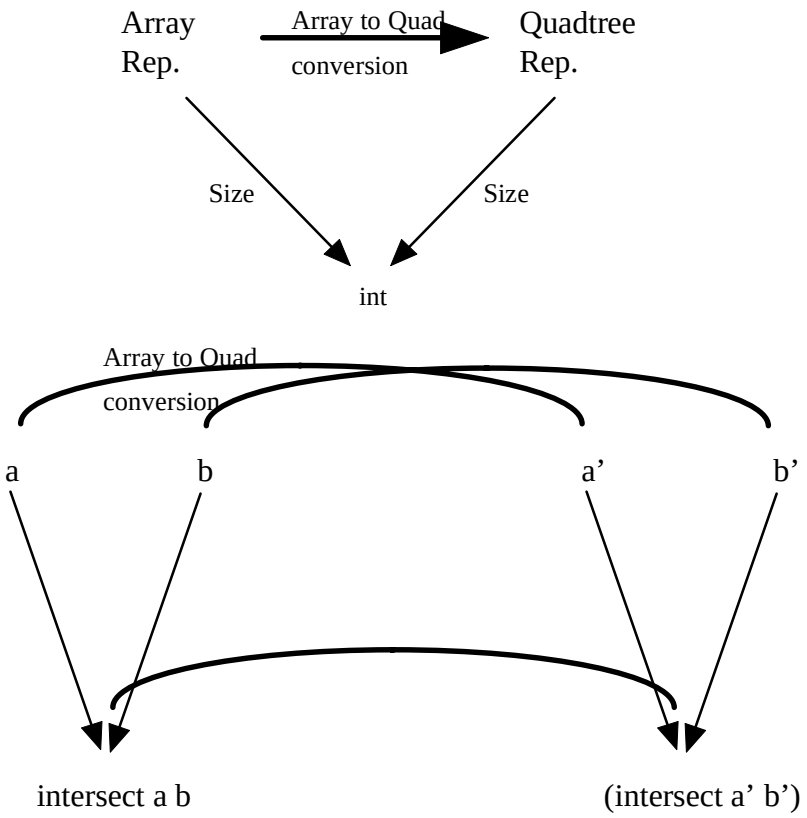

Figure 3 - Commutative diagram for size and intersection

This can be demonstrated inspecting the code.

5 The Essence of Hierarchical Reasoning

Hierarchical reasoning is based on the economic principle to use the least detailed representation to answer a question. Assuming that all data are inherently somewhat imprecise, and most tasks need only answers which are good enough for the decision at hand, it is sufficient to compute a result which is sufficiently precise. Computing more detail is a waste as it does not contribute to the solution.

Human behaviour follows nearly always this pattern and deviations are ridiculed: the engineer, who computes the diameter of a pencil to 15 places after the decimal point; the accountant, who declares the gross income of a company as $13'435'342'677.42 etc. all demonstrate some basic incomprehension of the limits to measurements of real world objects. The lay person marvels at the precise measurements possible today, when the length of the year can be determined down to very small fractions of a second, or the distance from Europe to the USA measured with an accuracy better than one centimeter. Standard measurement practice by lay persons seems to yield 1/1000, professionals achieve 1 ppm and specialists can push precision even further (to 10**-12).

Hierarchical reasoning is based on data structures which represent the objects of interest with an appropriate margin of error and level of detail commensurate with it. For figures, one expects a small number of significant digits (4 to 6): the number of inhabitants of a small town is given (and accepted) as 221, but everybody knows that the population figures for a nation (e.g., the result of the US census) have a margin of error. Therefore, multiple representations of objects at different levels of resolution and - implied - with different error margins are constructed and used.

Information is needed to make decisions. Human spatial reasoning deduces a result from a given information and implicitly assesses the quality of this result. If the quality of the result is sufficient that the decision can be taken with acceptable risk, then there is no need to collect more detailed information and to deduce a better result.

5.1 Array Representation

From a given array of resels at maximal resolution a sequence of coarser representations are constructed by grouping together four resels and giving them a new value: 'inside' if all four smaller resels are inside, 'outside' if they are all outside. For the mixed case, we can either employ a majority rule (3 resels inside give 'inside' for the aggregate, 3 resels outside give 'outside' for the aggregate), but to decide on the split case (2 inside, 2 outside) is difficult. Any decision will bias the result one way or the other.

The solution selected here is an 'all' rule: only all inside or all outside cases are recorded as inside or outside in the coarser representation and all mixed cases marked as such. Resels can then have 3 values, inside, outside or mixed (Figure 4).

```
data Pixel3 = I3 | O3 | M3
```

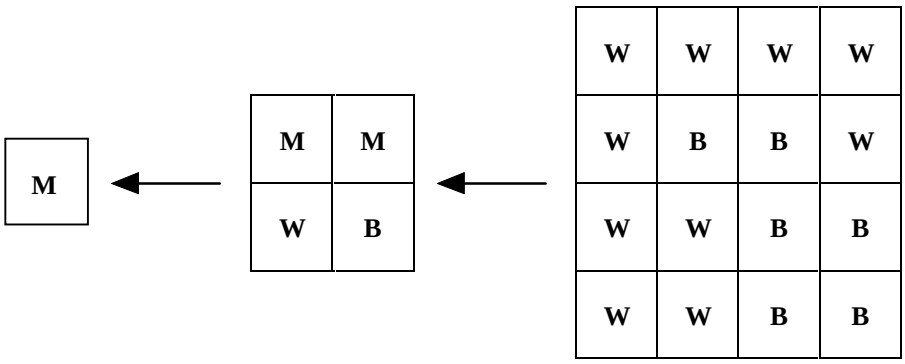

Figure 4 - Three levels of aggregation of resel array

5.2 Hierarchical Computation of Area and Intersection

The computation for the area and the intersection values in a hierarchical algorithm must include the computation for the error bounds. A general type of *value with error* pair is introduced. The quality of the result is here - for simplification - represented with the area of uncertainty, i.e. the area which might possibly add to the area count already fixed, or the area of uncertainty in which an intersection could possibly happen.

```
data VE v e = VE v e
```

For these 'values with errors' the standard operations (+) or 'and' and 'or' must be defined, carrying forward the errors (in defining the operations, care must be taken to assure that commutativity, associativity etc. of these operations is preserved)[2].

[2]
```
instance Num v => Num (VE v Int) where
     (VE v1 e1) + (VE v2 e2) = VE (v1 + v2) (e1 + e2)
instance (ZeroOne (VE v Int), Bools v) => Bools (VE v Int) where
     (VE v1 e1) &&& (VE v2 e2) = VE (v1 &&& v2) (max e1 e2)
     (VE v1 e1) ||| (VE v2 e2) = VE (v1 ||| v2) (max e1 e2)
     not1 (VE v1 e1) = VE (not1 v1) e1
```

5.2.1 Area Computation

The computations for area and intersection are mapped back to the corresponding computations for the pixels with 2 values:

```
instance Areas Pixel3 (VE Int Int) where
    area M3 = VE 0 1 -- a mixed resel contributes only to the error
    area p =  VE (area p) 0 -- a black or white resel contributes
nothing to the error
```

5.2.2 Intersection Computation

The intersection operations can also be performed on coarser representations. If the two regions are not intersecting on the coarser representation, they are also not intersecting on the finer one. There are, however, cases where we cannot decide if two resels intersect on the coarser level; this is the case for all combinations with a mixed resel. The computation of the intersection therefore yields three values:`true` and `false` (for computations with resels with values of inside or outside) or`maybe` (if one of the two resels is a mixed case); the three-valued logic proposed by Lukasiewicz is applicable (Sinowjew 1968) with the logical operators as given in Figure 5. The error bound on the result is computed as the maximal area where an overlap could occur.

```
instance Regions Pixel3 (VE Bool3 Int) where
    intersect a b = VE v (if v==Maybe then 1 else 0)
            where v = (intersect a b)
```

When the test for intersection, defined for the combination of two resels, is spread over the tree, the partial results must be connected logically. The interpretation is:

- if there is any certain overlap, the intersection test yields`true`;
- if there is no certain overlap, but any maybe value, the intersection test yields `maybe` (the number of maybe resels gives abound for the error);
- if all resels are ‘no overlap’, the intersection test yields`false`.

A	B	A and B	A or B
T	T	T	T
T	M	M	T
T	F	F	T
M	T	M	T
M	M	M	M
M	F	F	M
F	T	F	T
F	M	F	M
F	F	F	F

A	not A
T	F
M	M
F	T

Figure 5 - Truth tables for ‘and’, ‘or’ and ‘not’ in the three-valued logic of Lukasiewicz

5.3 Quadtrees for Hierarchical Reasoning

Quadtrees are efficient hierarchical data structure to replace resel arrays. Standard quadtree processing is more efficient as it exploits spatial autocorrelation and computes always correct results. Hierarchical reasoning computes results with error assessments and stops processing when an acceptable result is achieved. A quadtree must be inspected only as deep as the contribution of the details is necessary to get a sufficiently precise result. Conceptually, one can imagine that the quadtree leaves are cut away at the level of detail which is of no further interest (Figure 6):

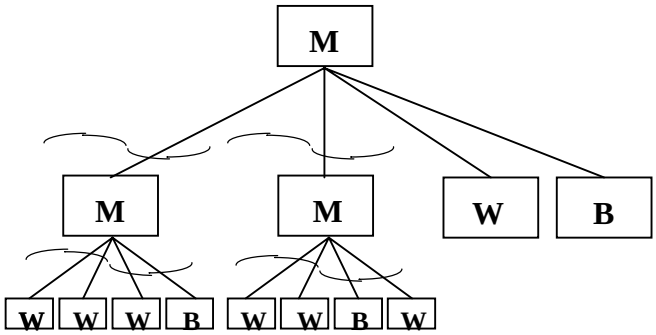

Figure 6 - Quadtree cut at levels

This can be seen as a 'coarsening' operation on the quad resels, which is replacing inner nodes at the 'cut-off' level with leaves of value M (Samet 1989a, p. 326 with additional references). Then area size and intersection operations are performed on this coarser quadtree. The coarsening of the quadtree and performing the operation could be integrated in a single operation, but this is a technical detail of optimisation.

6 Definition of Spatial Hierarchical Reasoning

Spatial hierarchical reasoning assumes that the given values have a certain error and that results need only to be computed at a level of certainty appropriate for the task at hand. Both are often only implied assumptions and need to be made explicit for the formulation of hierarchical reasoning algorithms. The estimate of the error in the values given is associated with the coarseness level of their representation (e.g., how many digital places given), but must be introduced quantitatively. Similarly, the tolerated error in the result must be given in the same measurements. Here it is assumed to be directly related to the spatial resolution of the resels.

To apply spatial hierarchical reasoning, the following is necessary:

- an algorithm f to work on a representation of the facts at a given level of detail (in our case a resel array of a given resolution) producing a value v;
- an algorithm f' to work on the representation r_i and producing an estimate of the error e_i on the result of applying f to the representation r_i, yielding v_i (above we have merged f and f' in single computations for area (with area uncertainty) and intersection (with intersection uncertainty));
- a filter c to produce a sequence of representations r_i, such that r_{i-1} is coarser than r_i (observe the ordering of the representations, which proceeds from the

coarsest to the finest: r_0 is the coarsest representation, r_i is the finest; therefore $r_{i-1} = c(r_i)$).

The computation at one level of detail must yield a result, which is compatible with the result of the next level with more detail and, by transitive extension, eventually compatible with the true result. The approximations must monotonically improve towards the correct result. This imposes a number of conditions on the coarsening operation and the spatial reasoning operation.

6.1 Formal Conditions

Given a sequence of representations of the same situation with increasingly more detail:

$r_1, r_2 \ldots r_i \ldots r_m$

where r_1 is the coarsest and r_m is the most detailed representation. The coarsening operation c produces r_{i-1} from r_i:

$r_{i-1} = c(r_i)$

The operation of interest *f* applied to each element of this sequence of r_i produces a sequence of values v_i and with the corresponding error estimation function *f'* a sequence of error estimates e_i.

$v_1, v_2 \ldots v_m$ *with* $v_1 = f(r_i)$

$e_1, e_2 \ldots e_m$ *with* $e_i = f'(r_i)$

6.1.1 Conversion Towards Correct Result

The computations must converge towards the correct result. This requires that the error bounds get more and more restraining. This means that the sequence $e_1, e_2 \ldots e_m$ must monotonically decrease, i.e. $e_i \geq e_{i+1}$.

This gives the condition

$$f'(c(r_i)) \geq f'(r_i)$$

for the functions *f'* and the coarsening function *c* (because $e_i = f'r$ and $e_{i+1} = f'(r_{i+1}) = f'(c(r_i))$).

This is the case for the coarsening function on arrays and the computation of error bounds as the sum of the 'mixed' pixels: the coarsening operation produces a larger and larger area of mixed pixels (if one of the four pixels is mixed, the result of the coarsening is mixed).

6.1.2 Non-Contradiction of More Detailed Values

The value reported by the calculation with a more detailed representation must not contradict the value achieved with a less detailed representation. For Boolean values resulting from the intersection calculation, once a value of `True` or `False` is achieved, all computations with more detail must produce the same value. Only for a value of v_i = Maybe, the value v_{i+1} can be either True or False.

For the calculation of area, the value e_i is the sum of the inside pixels, the value v_i is the sum of the mixed pixels. The total area can vary between v_i and $v_i + e_i$. The interval resulting from a more detailed computation with representation r_j (with $j > i$)

$[v_j, v_j + e_j]$ must be inside the interval resulting from the detail level r_i, which is $[v_i, v_i + e_i]$. Simple computations give the conditions for the operations.

6.1.3 Termination

The computation of the values in the sequence progresses till a value v_i is found, for which the error bound e_i is smaller than the error which can be tolerated for the decision to be made. No further computation is necessary. The sequence of error values must monotonically decrease and reach ultimately *0*.

This is efficient as it excludes all computations with detail which are not significantly contributing to the solution of the problem.

7 Optimization: Incremental Spatial Reasoning

Spatial reasoning as broadly defined above is computing the value for each representation from scratch, repeating for all areas where there is no additional detail, the same calculation for each level, achieving the same results. This is clearly inefficient and can be improved using an incremental approach: instead of recomputing the value for the representation r_i from scratch, the value obtained from r_{i-1} is used and only the increments Δv_i and Δe_i are computed and added to v_{i-1} and e_{i-1}.

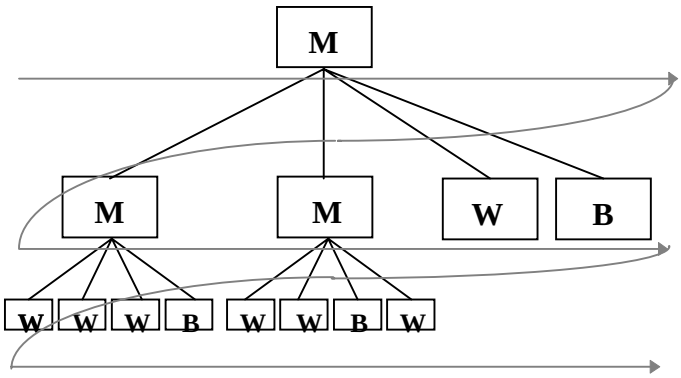

Figure 7 - Incremental processing: arrows mark the flow of computation

To compute the increment, only areas which have details must be inspected; in the representation used here, only mixed resels must be visited. Calculating the area, only these add to the area count and decrease the error level. When calculating the intersection and the overall result is `maybe`, only resels which yielded `maybe` must be inspected on the more detailed level to see if a `true` or `false` result can be obtained.

The same incremental concept is applied to the data structure resulting in hierarchical incremental data structures, where a finer level of detail contains only the increment in detail and does not repeat the representation of the areas, where the coarser and finer representation agree. Such data structures have, in the best case, the same size as non-hierarchical ones and give all the advantages of multiple-representations. An incremental variant of a quadtree will serve as an example.

7.1 Incremental Hierarchical Reasoning

Incremental hierarchical spatial reasoning is a special case of hierarchical spatial reasoning. It requires that the coarsening filter does not construct a series of increasingly detailed structures $r_1 \ldots r_m$, but a sequence of increments, such that $i_1 = r_1$ and $r_j = r_{j-1} + i_j$.

The algorithm *f* and *f'* for spatial reasoning are transformed to work on the i_1 and produce Δv_i and Δe_i, such that

$$v_j = v_{j-1} + \Delta v_j$$

with $v_j = \bar{f}(i_j)$, and

$$e_j = e_{j-1} + \Delta e_j$$

with $\Delta e_j = \overline{f}'(i_j)$.

7.2 Incremental Quadtree

Quadtrees are pointer structures and are most efficiently searched 'depth first'. For an incremental algorithm, we need a 'breadthfirst' search. This is effectively searching the quadtree with detail level 0 and then process the additional nodes that needed expansion to get to the next more detailed representation level by level.

A regular quadtree can be transformed into a structure which gives the nodes for each level (in left to right order) - this is the result of single breadth first traversal (and can be done in linear time in the number of nodes). The incremental quadtree can be represented as a list of levels, each level as a list of the nodes (coded as above as I, O or M). This transformation can be reversed, as all details are preserved (the transformation is lossless). The number of nodes remains the same and storage is very compact as no pointers are stored (this is closely related to the linearized, pointerless quadtree (Samet 1989a, p. 55) resulting from a depth-first traversal).

Applying the size or intersection test to each level gives the increment this level contributes to the improvement. If we sum up these increments till we reach the desired level of quality in the result (i.e. an error smaller than acceptable), the desired efficient computation is achieved. The amount of computation is exactly the same as in a regular quadtree for the same depth of detail. With the regular quadtree algorithm the processing would not have progressed in increasing the level of detail and stopped, when a sufficient result achieved. It would have proceeded till all details are inspected to produce the unnecessary precise result.

8 Open Questions

The major open problem is to find general rules to decide which spatial objects and operations can be brought into this framework. How does it apply to linear or point features? How does it extend to representations with irregular tesselations (vector data model)?

8.1 Hierarchical Map Algebra

A large part of areal spatial reasoning is included in the concept of the map algebra. It is the logical framework for many of the widely available GIS software systems. In principle, all these operations can be brought into the framework of hierarchical spatial reasoning, because any operation can be applied to the representation expanded as an array of resels (of the given level). It can be expected, that more efficient solutions are possible, using the results from research in efficient algorithm in quadtrees (Mark and Lauzon 1985).

The difficult task is to determine the corresponding functions to compute errors for spatial analysis functions. This is a generally useful exercise, which yields insight into the tasks and analytical functions.

8.2 Hierarchical Reasoning on Irregular Tesselations

The discussion here was motivated by the quadtree organization of a regular tessellation (raster). Much effort has been invested in irregular tesselations, as they represent an irregular situation with more resolution and less storage cost. Efforts to construct hierarchies of irregular tesselations have been started (Bruegger and Egenhofer 1989; Bruegger and Frank 1990; De Floriani and Puppo 1992; Puppo and Dettori 1995).

The logical framework presented here does not depend on the type of spatial subdivision, and works equally with irregular tessellations if they form an inclusion hierarchy, even if there is an irregular fan out.

8.3 Hierarchical Reasoning for Line-or Point-based Spatial Problems

Car has studied hierarchical reasoning for a hierarchically structured network and used the shortest path problem (Car to appear). Her definition of a hierarchical reasoning process is very similar, but in the framework of point- and line-based objects (Car 1996).

For point- and line-based spatial problems, the definition above would not immediately work. The hierarchy of a e.g., road network is not based on the transitive inclusion hierarchy of (spatial) containers. Depending on the applications, for example, a hierarchy can be established by defining for each node a series of ‘coarsened nodes’, such that the coarsened node is the nearest node to a given node on the next upper level. Then the hierarchical reasoning is applied with a sequence of coarsening of the start and goal node. It appears that this algorithm can be made incremental, but only as a heuristic; correct only if some restrictions on the graph are fulfilled.

9 Conclusions

“Hierarchy is fundamental to human cognition” (Langacker 1987, p. 310). Humans apply hierarchical concepts to spatial situations very often. It is one of the major conceptual tools to structure the infinite levels of detail in our spatial environment. The spatial inclusion hierarchy is used to make reasoning more efficient: results are deduced at the coarsest level of detail to reduce the amount of facts to be considered and areas, for which it is possible to quickly exclude that they contribute to the solution, are discarded and not further explored.

Spatial hierarchical reasoning proceeds in steps of increasing resolution and produces increasingly better approximations to the correct result, till an approximation is found which is ‘good enough’ to fulfil the requirements of the task at hand. (In real-time applications, the same method can be used to compute a series of increasingly better approximations to fulfil stringent requirements on response: it is sometimes more important to have a first approximation quickly, then to wait till the correct result becomes available too late.)

Spatial hierarchical reasoning is thus the combination of

- a data structure which provides increasingly more detail and allows to compute an error bound on the approximation, and

- an algorithm which computes the desired value and an error bound for it (this is generally recommended for spatial algorithms, but seldom done for most computation in a geographic information system, where input data are always somewhat erroneous).

Quadtrees are a well known, efficient method to support this reasoning approach. Standard quadtree algorithms produce correct results with minimal effort. We have shown here, how quadtrees can be used for a hierarchical processing, which stops when the quality of the result is sufficient for the decision to be made. In spatial information systems, which cover multiple levels of detail, this can result in large performance improvements, as processing of a complete level of detail is avoided.

Spatial hierarchical reasoning can be optimized if the data structure and the algorithm permit incremental processing. Incremental spatial hierarchical reasoning proceeds to compute a result and then computes a series of increments, which improve the previous result. This makes computation as efficient as a straightforward approach, except that it produces intermediate values, which are useful, and can be stopped, when the desired error bound has been achieved. This can also be used for very demanding real time application, where improving approximations are computed to guide reactions to a changing environment.

In this framework, general rules when and how spatial hierarchical reasoning is possible, were given. For each reasoning task, a computation of error bounds must be possible. The interaction of the method to deduce the less detailed representation and the computation of error bounds must correspond, such that

- the computation converges towards the correct result,
- more detailed values do not contradict previously computed values, and
- the computation terminates.

It is expected, that most of the standard spatial operators (e.g., from the map algebra (Tomlin 1983a; Tomlin 1983b; Tomlin 1989)) can be translated into the hierarchical reasoning framework and that even incremental algorithms can be found.

References

Abler, R., et al.. 1971. *Spatial Organization - The Geographer's View of the World.* Englewood Cliffs, NJ, USA: Prentice Hall.

Ballard, D. H. 1981. Strip Trees: A Hierarchical Representation for Curves. *ACM Comm.* 24(5):310-321.

Bruegger, B. P., and M. Egenhofer. 1989. Hierarchies over Topological Cells for Databases. In Proceedings of GIS/LIS '89, at Orlando, FL.

Bruegger, B. P., and A. U. Frank. 1990. Hierarchical extensions of topological datastructures (P301.2). In Proceedings of FIG XIX Congress, June 10-19, 1990, at Helsinki, Finland.

Burrough, P. A., and A. U. Frank, eds. 1996. *Geographic Objects with Indeterminate Boundaries.* London: Taylor & Francis.

Buttenfield, B. P. 1993. Multiple Representations - Closing Report. Buffalo: State University of New York at Buffalo.

Buttenfield, B. P., and J. S. Delotto. 1989. Multiple Representations: Report on the Specialist Meeting - Initiative 3: National Center for Geographic Information and Analysis; Santa Barbara, CA.

Car, A. 1996. *Hierarchical Spatial Reasoning: Theoretical Consideration and its Application to Modeling Wayfinding*. Doctoral Thesis. *GeoInfo Series, Vol.10*. Department of Geoinformation, Technical University Vienna.

Car, A. to appear. Hierarchical Wayfinding - A Model and its Formalization. Proceedings of ESF - GISDATA Summer Institute (1996), at Berlin.

Car, A., and A. U. Frank. 1994a. General Principles of Hierarchical Spatial Reasoning - The Case of Wayfinding. In Proceedings of Sixth International Symposium on Spatial Data Handling, SDH'94, at Edinburgh, Scotland.

Car, A., and A. U. Frank. 1994b. Modelling a Hierarchy of Space Applied to Large Road Networks. In *IGIS'94: Geographic Information Systems. Proceedings of International Workshop on Advanced Research in GIS in Ascona, Switzerland*, Berlin-Heidelberg: Springer-Verlag.

Cohn, A.G. 1995. A Hierarchical Representation of Qualitative Shape Based on Connection and Convexity. In *Spatial Information Theory-A Theoretical Basis for GIS*, - Proceedings of Int. Conference COSIT '95, edited by A. U. Frank and W. Kuhn. Berlin-Heidelberg: Springer-Verlag.

Davis, E. 1990. *Representation of Commonsense Knowledge.* San Mateo, CA: Morgan Kaufmann Publishers.

De Floriani, L., and E. Puppo. 1992. A Hierarchical Triangle-Based Model for Terrain Description. In *Theories and Methods of Spatio-Temporal Reasoning in Geographic Space*, edited by A. U. Frank, I. Campari and U. Formentini. Berlin-Heidelberg: Springer-Verlag.

Dutton, G. 1993. Scale change via hierarchical coarsening: cartographic properties of Quaternary Triangular Meshes. In Proceedings of 16th International Cartographic Conference, at Koeln, Germany.

Fotheringam, A.S. 1992. Encoding Spatial Information: The Evidence for Hierarchical Processing. In *Theories and Methods of Spatio-Temporal Reasoning in Geographic Space*, edited by A. U. Frank, I. Campari and U. Formentini. Berlin-Heidelberg: Springer-Verlag.

Fraser, J. T., ed. 1981. *The Voices of Time*. Second Edition. Amherst: The University of Massachusetts Press.

Freksa, C. 1991. Qualitative Spatial Reasoning. In *Cognitive and Linguistic Aspects of Geographic Space*, edited by D. M. Mark and A. U. Frank. Dordrecht: Kluwer Academic Publishers.

Freksa, C. 1992. Using Orientation Information for Qualitative Spatial Reasoning. In *Theories and Methods of Spatio-Temporal Reasoning in Geographic Space*, edited by A. U. Frank, I. Campari and U. Formentini. Berlin-Heidelberg: Springer-Verlag.

Golledge, R. G. 1992. Do People Understand Spatial Concepts: The Case of First-Order Primitives. In *Theories and Methods of Spatio-Temporal Reasoning in Geographic Space*, edited by A. U. Frank, I. Campari and U. Formentini. Berlin-Heidelberg: Springer-Verlag.

Goodchild, M. F., and S. Gopal, eds. 1989. *Accuracy of Spatial Databases*. London: Taylor & Francis.

Goodchild, M. F., and Yang Shiren. 1990. A Hierarchical Data Structure for Global Geographic Information Systems. In Proceedings of 4th International Symposium on Spatial Data Handling, at Zurich, Switzerland.

Hirtle, S. C., and J. Jonides. 1985. Evidence of Hierarchies in Cognitive Maps. *Memory & Cognition* 13 (3):208-217.

Hudak, P., et al.. 1992. Report on the functional programming language Haskell, Version 1.2. *SIGPLAN Notices* 27.

Jones, C. B., and L. Q. Luo. 1994. Hierarchies and Objects in Deductive Spatial Databases. In Proceedings of Sixth International Symposium on Spatial Data Handling, SDH'94, at Edinburgh, Scotland.

Jones, M. P. 1991. An Introduction to Gofer. Technical Report: Yale University.

Jones, M. P. 1994. Gofer - Functional Programming Environment: Geoinformation, TU Vienna.

Lakoff, G., and M. Johnson. 1980. *Metaphors We Live By.* Chicago: University of Chicago Press.

Langacker, R. W. 1987. *Foundations of Cognitive Grammar*. Vol. 1 - Theoretical Prerequisites. Stanford, CA.: Stanford University Press.

Mark, D. M., and J. P. Lauzon. 1985. The space efficiency of quadtrees: an empirical examination including the effects of 2-dimensional run-encoding. *Geo-Processing* 2:367-383.

NCGIA. 1989. The Research Plan of the National Center for Geographic Information and Analysis. *IJGIS* 3(2):117-136.

Noronha, V. T. 1988. A Survey of Hierarchical Partitioning Methods for Vector Images. In Proceedings of Third International Symposium on Spatial Data Handling, at Sydney, Australia.

Overmars, M. H. 1983. *The Design of Dynamic Data Structures*., Vol. 156, *Lecture Notes in Computer Science.* Berlin-Heidelberg: Springer-Verlag.

Papadias, D. and J. I. Glasgow. 1991. A knowledge representation scheme for computational imagery. In Proceedings of Thirteenth Annual Meeting of the Cognitive Science Society, at Chicago, USA.

Puppo, E., and G. Dettori. 1995. Towards a Formal Model for Multiresolution Spatial Maps. In *Advances in Spatial Databases (Proceedings SSD'95)*, edited by E. M. J. and J. R. Herring. Berlin-Heidelberg: Springer-Verlag.

Rigaux, P., et al.. 1993. A Map Editing Kernel Implementation: Application to Multiple Scale Display. In *Spatial Information Theory: Theoretical Basis for GIS*, - Proceedings of Int. Conference COSIT '95, edited by A. U. Frank and I. Campari. Heidelberg-Berlin: Springer-Verlag.

Samet, H. 1989a. *Applications of Spatial Data Structures. Computer Graphics, Image Processing and GIS*. Reading, MA: Addison-Wesley.

Samet, H. 1989b. *The Design and Analysis of Spatial Data Structures*. Reading, MA: Addison-Wesley.

Sinowjew, A. A. 1968. *Über mehrwertige Logik*. Berlin: Deutscher Verlag der Wissenschaften.

Stevens, A., and P. Coupe. 1978. Distortions in judged spatial relations. *Cognitive Psychology* 10:422-437.

Timpf, S. to appear. A multi-scale data structure for cartographic objects. In *Geographic Information Research: Bridging the Atlantic*, edited by M. Craglia and H. Couclelis. London: Taylor & Francis.

Timpf, S., and A. U. Frank. 1995. A Multi-Scale Dag for Cartographic Objects. In Proceedings of ACSM/ASPRS, at Charlotte, NC

Timpf, S., et al.. 1992. A Conceptual Model of Wayfinding Using Multiple Levels of Abstractions. In *Theories and Methods of Spatio-Temporal Reasoning in Geographic Space*, edited by A. U. Frank, I. Campari and U. Formentini. Berlin-Heidelberg: Springer-Verlag.

Tobler, W.. 1995. The Resel Based GIS. *IJGIS* 9 (1):95-100.

Tomlin, C.D. 1983a. Digital Cartographic Modeling Techniques in Environmental Planning. Doctoral Thesis., Yale University.

Tomlin, C. D. 1983b. A Map Algebra. In Proceedings of Harvard Computer Graphics Conference, at Cambridge, Mass.

Tomlin, C. D. 1989. *Geographic Information System and Cartographic Modeling*. New York: Prentice Hall.

Voisard, A., and H. Schweppe. 1994. A Multilayer Approach to the Open GIS Design Problem. Second ACM-GIS Workshop, at New York.

Whigham, P. 1993. Hierarchies of Space and Time. In *Spatial Information Theory: Theoretical Basis for GIS*, - Proceedings of European Conference COSIT '93, edited by A. U. Frank and I. Campari. Berlin-Heidelberg: Springer-Verlag.