Proceedings of the 6th AGILE
April 24th-26th, 2003 – Lyon, France

A SYSTEM ARCHITECTURE TO IMPLEMENT A UNIFIED APPROACH FOR SPATIAL OBJECT MODELLING AND MAP ANALYSIS BASED ON 2ND ORDER MANY-SORTED LANGUAGE

Oscar Luiz Monteiro de Farias¹, D. Sc., Sueli Bandeira Teixeira Mendes¹, Ph. D., Margareth Simões Penello Meirelles¹, Ph. D. Luis Fernando Bueno¹,², M.Sc. student

¹ Universidade do Estado do Rio de Janeiro - Centro de Tecnologia e Ciências - Faculdade de Engenharia, Departamento de Engenharia de Sistemas e Computação, Rua São Francisco Xavier, 524, bloco D, sala 5017, Rio de Janeiro, Brasil, CEP 20550-013, tel.: (+55) (21) 587-7442, fax: (+55) (21) 587-7374

² Universidade Luterana do Brasil - Unidade de Gestão de Conhecimento, Computação - Faculdade de Informática, Campi Carazinho, Rodovia BR 285 km 335 - CEP 99500000, tel: (54) 3291111, Fax: (54) 3291010

{oscar, smendes, maggie}@eng.uerj.br, lfbueno@annex.com.br

Abstract. 2⁰ Order Many-Sorted Language is introduced formally in [1] as an algebraic structure and, as such, yields an adequate tool for formalizing an extension of Peano Algebra for spatial object modeling, spatial data base information retrieval and manipulation, and also for map analysis. This tool provides a unified treatment for several classes of operations realized among maps, including those presented in Boolean Logic Models, Fuzzy Logic Models and Bayesian Probability Models. The present paper proposes a System Architecture for Implementing a Unified Approach for Spatial Object Modelling and Map Analysis Based on 2⁰ Order Many-Sorted Language. Capital aspects of this architecture are: i) the internal representation of spatial objects, basically based in quadtrees and Peano relations and stored in some spatial data base; ii) a compiler, that translates declarative sentences written in the 2⁰ Order Language into algorithms that are further executed against the internal representation of the spatial objects; iii) a Graphical User Interface, that shows to the users spatial objects related as in a map, and allows operations in these spatial objects and also Map Analysis, in which operations are realized between maps furnishing, as output, a new map.

1. INTRODUCTION

Farias and Mendes [1] proposed a Unified Approach for Spatial Object Modelling and Map Analysis Based on 2⁰ Order Many-Sorted Language. In that approach spatial objects were represented with the aid of quadtrees and Peano relations. Operations on spatial objects comprised traditional ones like: set operations, geometric operations, relational operations, and also analytical operations on single maps and on multiple maps, typical of map analysis. The later ones could be: map reclassification, aggregation operations over the attributes of the objects in a map, two-map overlays, multiple-maps operations based on Boolean Logic, Fuzzy Logic or Bayesian methods [2]. All the traditional operations on spatial
objects and also that operations related with map analysis were incorporated through the use of a Peano Algebra Extension. For formalizing this, Farias and Mendes [1] proposed a many-sorted $2^{nd}$ order language, since it is necessary to allow variable quantification over predicates.

In this paper we present a System Architecture for Implementing the Unified Approach for Spatial Object Modelling and Map Analysis Based on $2^{nd}$ Order Many-Sorted Language proposed by Farias and Mendes [1].

First we present some background material related to Space Filling Curves, Peano Curves, Quadtrees, Peano Relations and Spatial Modelling. After that (in 3), we show how we can realize several operations on spatial objects, including some map analysis operations. These operations are carried based on the spatial model presented in 2.2. Then we formalize a Peano Algebra Extension through a $2^{nd}$ Order Many-Sorted Language, described (in 5). Finally (in 6) we illustrate the translation of the algorithms shown in 3. In section 7 we summarize the architecture for implementing the system. Finally, we ending the paper with the traditional conclusion.

2. MODELING SPATIAL OBJECTS

2.1 Space Filling Curves

Space filling curves (Fig. 1) are an attempt to represent n-dimensional systems by a one-dimensional system. The basic idea is to build a curve that travels by all points in a n-dimensional space. In the case of two-dimensions, we’ll need to find curves that pass for all points in a given plane. Of course, according to the Euclidian geometry, this is not theoretically possible, if we define a point as a zero-dimensional object. However, we can have a solution for this problem if we think of a point as a two-dimensional object, a square which sides tend towards zero.

![Fig. 1 Space Filling Curves](image)

With the help of fractal geometry, it is now possible to find a curve that goes through all these small squares and fills completely a two-dimensional space. This curve has a certain width that will also tend to zero as the sides of the squares that it traverses tend also to zero. A “good” space filling curve should observe some properties such that: i) the curve must pass only once to every point in the multi-dimensional space; ii) the curve must correspond to a bijective mapping from a one- to a multi-dimensional space, etc [3].

Two well-known curves: the Peano-curve and Hilbert-curve meet most of the conditions for a “good” space filling curve. But the Hilbert curve does not provide an easy way to retrieve neighbours points in the space and is not stable when we operate through
different scales. The computation of Hilbert-keys is also more difficult than the computation of Peano-keys. These keys reflect a bijective map of the n coordinates \((x_1, x_2, \ldots, x_n)\) on a single coordinate: the so-called Peano-key (Pk) or Hilbert-key (Hk). The initial steps for building the Peano curve are show in Fig. 2

![Fig. 2 Peano Curves](image)

### 2.2 Quadtrees, Peano-Relations and Spatial Representation

We can use quadtree to approximate an arbitrary object successively by a set of blocks at different levels, where each of these blocks is the result of a quadrant recursive subdivision [4]. The grey object at left in Fig. 3, for example, can be represented by blocks of different sizes belonging to three distinct levels of quadrant subdivisions. This process can be applied to any object, until a predefined resolution level is reached. This could be, for example, the resolution of a given monitor. Now, we can use the set of pairs (Peano-keys, side\_length) of all the blocks that comprises the object as a spatial representation of the object. Indeed, we can represent all the objects in a two-dimensional space, for example, by the so called Peano relation PR(Object\_id, PK1, side\_length). The Peano relation is a kind of intensional-extensional representation for spatial objects. Its meaning is that an object is represented by the union of several tuples. Each of these tuples represents a square-region of the two-dimensional space. This region is univocally determined by the Peano-key associated to the square and the side-length of the square. The tuples representative of the yellow object at left of figure 3 would be:

- PR (A, 12, 2, At1, ..., At3)
- PR (A, 35, 1, At1, ..., At3)
- PR (A, 36, 2, At1, ..., At3)
- PR (A, 44, 1, At1, ..., At3)
- PR (A, 48, 4, At1, ..., At3)

Where At1, ..., Atn, is a list of attributes associated to the object.
In this paper we will use the run-length-encoding representation for Peano-relations, that is: PR(Object_id, PK1, PK2, At1, ..., At_i). In this alternative Peano relation, each object can be thought as the union of several tuples as before, but the square-region associated with a tuple means that this square occupies the region of the two-dimensional space with Peano-keys (PK) in the interval [PK1, PK2]. That is PK1≥PK1 and PK<PK2.

3. A UNIFIED APPROACH FOR MODELLING SPATIAL OBJECTS AND ALSO FOR THE OPERATIONS REALIZED ON THEM

Our goal in this paper is to use Peano relations and an extension of Peano-Algebra, in order to represent spatial objects and realize several different types of operations on them. These operations could be:

- Set operations (intersection, difference, union);
- Geometric operations (translation, rotation, scalling, symmetry, replication, simplification, window extraction);
- Relational operations (geometric projection, Peano Join);
- Spatial queries (query about topology);
- The traditional thematic queries (over the object attributes);
- Operations over two or more maps (binary evidence maps, index overlay with multi-class maps, Boolean models, Fuzzy Logic Models and Bayesian Models).

Due to space restrictions, we'll show here only some examples, but we hope the operations in these examples will give us some insight in the expressiveness and power of the extension to the Peano-Algebra proposed here. First we'll explain the realization of some operations based on Peano relations, and the we'll propose one particular Peano Algebra extension to generalize our approach.

First of all let us consider a Peano relation MAP (Object_id, PK1, PK2, At1, ..., At_i) that represents a set of objects spatially related, as in a two-dimensional MAP. Two objects “A” and “B” would be represented generically by the following set of “n” and “m” tuples in MAP:

MAP (A, PK1, PK2, At1, ..., At_i)  
.........  
MAP (A, PK1, PK2, At1, ..., At_i)  
.........  
MAP (Object_B, PK1, PK2, At1, ..., At_i)  
.........  
MAP (Object_B, PK1, PK2, At1, ..., At_i)
A) Suppose we want to make a map reclassification based on attributes of the existing classes [5]. A map can be thought as \( n \) objects (or \( n \) classes) related in a two-dimensional space. We could represent the map by the Peano relation MAP (Class_Id, PK1, PK2, At_1, ..., At_n). In the general case we would have \( n \) classes: Class_1, Class_2, ..., Class_n. The reclassification operation on a map could be regarded as a function \( f: A \rightarrow C \), where \( A \) stands for the set of attributes, and \( C \) for the set describing the new classes of the map. So the new map would be: MAP (f(At_1, ..., At_n))Class_Id, PK1, PK2, At_1, ..., At_n), that is, \( f \) is restricted to each Class. On reality, this map would yet need to be put in the third conformance level [3], because identical classes could be dispersed in the space.

B) Now we want to analyse two-map overlays, a powerful tool for examining the spatial patterns that arise by the interaction between two maps [5]. In this case the goal is to combine the input maps according to a set of rules (the map model) that determines for each location the class of the output map from the classes of the input maps, that is, in each MAP we have a function \( f: Class_A \times Class_B \rightarrow New\_Class \), where Class_A stands for the set of classes in MAP_A, Class_B for the set of classes in MAP_B, and New_Class is the result of the application of the function \( f \). The two maps could be described by the following Peano relations: MAP_A (Class_Id_A, PK1, PK2, At_1, ..., At_n) and MAP_B (Class_Id_B, PK1, PK2, At_1, ..., At_n). The following algorithm could do the desired reclassification:

\[
\text{Result\_Map} = \emptyset;
\text{for each Class\_Id\_A in MAP\_A do:}
    \text{for each Class\_Id in MAP\_B do:}
        \text{TEMP=PEANO\_JOIN (Class\_Id\_A, Class\_Id\_B);}
        \text{//now, for each location, we calculate the output class}
        \text{Result\_MAP=TEMP (f(Class\_Id\_A,}
            \text{Class\_Id\_B), PK1, PK2, At_1, ..., At_n) \cup Result\_Map;}
    \text{end for;}
\text{end for;}
\text{// After the loop the Result\_Map Peano relation must be put in the third \//conformity level.}
\]

We recall that the Peano Join between two Peano relations means that we keep tuples which have matching Peano keys in both Peano relations. In the above case the variable TEMP is also a PEANO relation

4. THE PROPOSED 2\textsuperscript{ND} ORDER MANY-SORTED LANGUAGE

For formalizing a Peano Algebra Extension we use a many-sorted 2\textsuperscript{nd} order language, since when talking about a Peano relation, for instance PR (Object_Id, PK1, PKF, At_1, ..., At_n), for certain operations we can not consider the arguments of the operations as variables of the same sort. What we are trying to do here is to use the same type of language we use to formalize data structures. The need of a second order language comes from the fact that is necessary to allow variable quantification over predicates.

First, we present some needed formalism and definitions [6], [7], [8]:

A) 2\textsuperscript{nd} Order Language \( L^2 \), of type \( \tau = (n_0^i, n_1^j, i \in I, j \in J, I, J \subseteq N) \).

B) variables \( v_0, v_1, v_2 \) : many-sorted individuals (type 0)

C) Variables \( P_0^i, P_1^j \), predicates (type 1) variables of rank \( i \in [0, n], n \in N \).
D) Sentential connectives: $\land, \lor, \rightarrow, \leftrightarrow$.
E) Quantifiers: $\forall, \exists$.
F) Non-logical constants:
   F.1) Predicate constants: $r_i \ i \in I$, of rank $n_i^0$.
   F.2) Operation constants: $f_j \ j \in J$, of rank $n_j^1$.
G) Let $V$ be the set of individual variables, $P_n$ the set of predicate variables of rank $n$, $n \in \mathbb{N}$, where $\mathbb{N}$ is the set of natural numbers.
H) Let $\text{Var} = \bigcup_{i=1}^{n} P_i$, be the set of all variables.
I) As in the case of first order language $L$, the set $T_{m2}$ of terms of $L_2$, is the smallest set $X$, such that:
   I.1) $V \subseteq X \subseteq E$, where $E$ stands for the set of all expressions;
   I.2) $e_0, e_1, \ldots, e_{n-1} \in X$, $n_i^1 = n$ implies that $f_i(e_0, e_1, \ldots, e_{n-1}) \in X$. That is, no type 1 variables occurs in terms.
J) Any expression of the form $t_i(t_0, \ldots, t_{n-1})$, $i \in I$, $n = n$, or $P^n_i(t_0, t_1, \ldots, t_n)$, $i \in N$, where $t_0, t_1, \ldots, t_n$, are terms, is called an atomic formula.
K) The $\text{AF}_m$ be the set of atomic formulas of $L_2$.
L) The set of formulas: $F_m$ is the subset of $X$, such that:
   L.1) $\text{AF}_m \subseteq X \subseteq E$;
   L.2) If $e_0, e_1 \in X$ then $\neg e_0 \lor e_1, e_0 \lor e_1, e_0 \rightarrow e_1, e_0 \leftrightarrow e_1 \in X$.
   L.3) If $e \in X, v \in \text{Var}$, then $\forall v, \exists v \in X$.
M) An algebraic structure for $L_2$, is just an algebraic structure of similarity type $\tau$, exactly as in first order languages: $\Gamma = (A, R, F)$, i.e. A, R, F. The intention is that individuals variables $v_i \in V$, range over elements of the domain $A$, while predicate variables $v_i^n \in V_n$ range over n-place relations $R: A^n \rightarrow \{T, F\}$ on $A$, each $n \in N$. Thus, for instance, $\forall v, (\ldots)$ means "for every truth-value $(T, F, \ldots)$", because a n-place relation is essentially just a truth-value. Similarly, $\exists v, (\ldots)$ means "for a subset $B \subseteq A(\ldots)$", because a 1-place relation $R: A^1 \rightarrow \{T, F\}$ on $A$ is just a subset of $A$. Finally, $\forall v, (\ldots)$ means "for every binary relation $R$ on $A (\ldots)$". To be precise, we define an assignment $\Phi$ over $\Gamma$ to be a function with domain $\text{Var}$, such that:
   i) For $v_i \in V$, $\Phi(v_i) \in A$, as in first order logic.
   ii) For $P^n_i \in V_n$, $\Phi(P^n_i): A^n \rightarrow \{T, F\}$, i.e., $\Phi(P^n_i)$ is an n-place relation on $A$.

For $t \in T_{m2}$ we define $\Phi(t)$ as in first order language.

We introduce new satisfaction clause for atomic formulas:
   i) $\Phi$ satisfies $P^n_i t_0, \ldots, t_{n-1}$ if and only if $\Phi(P^n_i)(\Phi(t_0), \ldots, \Phi(t_{n-1})) = T$.

We introduce also new quantifier clauses:
   ii) $\Phi$ satisfies $\forall P^n_i P$ if and only if for every $R: A^n \rightarrow \{T, F\}$, $\Phi(\overline{P^n_i}_R)$ satisfies $P$ (Here, $P$ is a non-atomic formula, and this particular notation means that $\Phi$ satisfies $P^n_i$ for each $R$).
iii) $\Phi$ satisfies $\exists P^n \phi$ if and only if at least one $R: A^n \rightarrow \{T, F\}$, $\phi(\overline{P^n}_R)$ satisfies $\phi$.

5. **TRANSLATION OF SOME OPERATIONS IN THE 2ND ORDER MANY-SORTED LANGUAGE**

We have chosen the operations presented in 3 to be expressed in the proposed 2nd Order Many-Sorted Language. Here we will consider that:

i) $=, >, <, \leq, \in$ are non-logical constants;

ii) $\text{MAX}(X, Y)$ and $\bullet$ are non-logical operators, where, for instance $A \bullet PK1$ selects the Peano-key of object $A$ that belongs to the Peano relation $\text{MAP}(\text{Class}\_\text{Id}, PK1, PK2, At_1, ..., At_t)$.

- **Operation presented in 3.A):**
  Here we introduce the notation $f_w(x_1, ..., x_n)$ to denote the function $f$ restrict to $w$.
  Then we’ll have:
  $\forall x \forall y (x = \text{MAP}(\text{Class}\_\text{Id}, PK1, PK2, At_1, ..., At_t) \land y = At_1, ..., At_t \rightarrow \text{MAP}(f_{\text{Class}\_\text{Id}(At_1, ..., At_t), PK1, PK2, At_1, ..., At_t}) \land y = At_1, ..., At_t)$.

- **Operation presented in 3.B):**
  Let define $f_i: \text{Class}\_A \times \text{Class}\_B \rightarrow \text{New}\_\text{Class}$ as the function that has as domain the set of classes of $\text{MAP}\_A$ Cartesian Product the set of classes of $\text{MAP}\_B$, and $\text{New}\_\text{Class}$, as the result of operation $f_w$ described above.
  $\forall x \forall y \forall s \forall t \forall w \forall z [x = \text{Class}\_\text{Id}\_A \land y = \text{MAP}\_A \land x \in y \land s = \text{Class}\_\text{Id}\_B \land t=\text{MAP}\_B \land s \in t \land w = \text{PEANO}_\text{JOIN}(\text{Class}\_\text{Id}\_A, \text{Class}\_\text{Id}\_B) \land z = \text{RESULT}\_\text{MAP} \rightarrow z = (f_w(\text{Class}\_\text{Id}\_A, \text{Class}\_\text{Id}\_B), PK1, PK2, At_1, ..., At_t)]$.

6. **SYSTEM ARCHITECTURE**

Fundamental in the system, is the internal representation of spatial objects, based on quadtrees and Peano relations. All spatial objects are treated as tuples and stored in data bases. A Graphical User Interface (GUI) is needed, in order the user can visualize spatial objects in a given context, like a map. In reality the system must be able to represent several maps together and to realize operations with these multiple maps (Map Analysis) using, for example, Boolean Logic, Fuzy Logic or Bayesian Probability Models. Theses Maps are presented like a stack in the upper part of the GUI (Fig.4), immediately below the toolbar. In this toolbar we could have special tools for rotating an object, zoom, print, export/import, etc. The lower part of the GUI enables the user to enter some declarative sentence in the 2nd Order Many-Sorted Language proposed. We’ll need also a compiler to scan the valid sentences in the proposed 2nd Order Many-Sorted Language, to make the correspondent lexical and syntactic analysis and to produce an executable code. This compiler will, besides other things: i) associate simple variables with spatial objects and predicate variables with the relationships existing among objects; ii) transform the declarative statements of the 2nd Order language in well structured algorithms; After the compiler phase, the system executes the algorithms. The final result, usually a map, will be showed at the upper part of the GUI. This very simple system architecture will allow a unified approach to both spatial object modeling and map analysis, including some sophisticated mathematical tools.
6.1 Parsing the 2nd Order Language Sentences

Declarative sentences written in the 2nd Order Language must be scanned, recognized as valid and executed by the system. A compiler module realizes all these tasks [9], [10]. The parsing is very simple and could be done in small and recursive piece of code, taking advantage of the recursive definition of the well-formed-formulas (wffs) of the language. This can be easily seen below, in the fragment of the Backus Normal Form for the language, concerning wffs (the last two definitions in the BNF below).

But, as we are interested in a first prototype, we can use one of the several lexical and syntactical analyzers tools free-for-use. Some of them are: JLex, JFlex, CUP, SableCC, Flex, Lex , Yacc, Bison e JavaCC. JavaCC (Java Compiler-Compiler) is a syntactic analyzer tool developed by SUN Microsystems and having as target the Java programmers community. It has been chosen, among other reasons, because the prototype is being developed in Java. Besides that, it is a powerful, flexible, and easy-to-use tool, which generates a compiled-code easy to understand. All we need is to insert the Language Grammar in the JavaCC specification files, in order to have the lexical and syntactical analyzer ready.

Some semantic analyzer procedures could also be inserted in the Java CC specification files, but is necessary to build additional procedures for completing a semantic analyzer.

\[ \forall x \forall y [x = \text{MAP}(\text{Class}_i, PK_1, PK_2, A_{t_1}, \ldots, A_{t_1}) \land y = A_{t_1}, \ldots, A_{t_1} \rightarrow \text{MAP}(f_{\text{Class}_i}(A_{t_1}, \ldots, A_{t_1}), PK_1, PK_2, A_{t_1}, \ldots, A_{t_1})] \]

Fig. 4 Graphical User Interface
Backus Normal Form for the 2nd Order Language

\[\text{BNF} := \{ | \} \]
\[\text{Sentential connective} := \rightarrow | \land | \lor | \neg \]
\[\text{Individual variable} := \text{<letter> [ <alphanumeric> ]} \]
\[\text{Alphanumeric} := \text{<letter>} | \text{<digit>} \]
\[\text{<digit>} := a | b | \ldots | z | A | B | \ldots | Z \]
\[\text{Integer} := \text{<digit>} | \text{<digit>} \text{[<integer>] \}
\[\text{MAP} := \text{<MAP_individual_variable>} \] \text{[Indicate map variables]}
\[\text{SO} := \text{SO} \text{<individual_variable>} \] \text{[Indicate spatial objects]}
\[\text{MAP_individual_variable} := \text{<individual_variable>} \]
\[\text{Attribute_List} := \text{<individual_variable>} \text{[<Attribute_List>] \}
\[\text{Predicate variable} := \text{X}_1^n \text{[X}_2^n \ldots \text{X}_m^n \ldots \]
\[\text{Function Symbols} := \text{<Boolean Function Symbol> | <Geometric Function Symbol> |}
\[\text{Relational Function Symbol>} \text{[Map Analysis Function Symbol]}
\[\text{Boolean Function Symbol} := \text{(SO), (SO)} \lor \text{((SO, (SO)) \}
\[\text{Fuzzy Logic Model} := \text{BLM ((<MAP_individual_variable>, <SO>),}
\[\text{Index Overlay Model} := \text{<Binary Evidence Maps> | <Multi-Class Maps>}
\[\text{Binary Evidence Maps} := \text{BEM ((<MAP_individual_variable>, <map_weight>),}
\[\text{Multi-Class Maps} := \text{MCM ((<MAP_individual_variable>, <map_weight>, (SO), <class_weight>))}
\[\text{Quantifier symbol} := \forall, \exists \]
\[\text{Constant symbol} := a_0, a_1, \ldots, a_n \]
\[\text{Predicate constant} := \text{Intersection (SO), (SO)} \text{Is_in (SO, (SO)) Adjacency (SO,}
\[\text{Only a few predicate constants are shown in the BNF. Here, } r_0^n | \text{r}_0^n | \ldots | r_m^n \text{ represent all the other predicate constants of the Language, for instance, those that describe topological relations among objects} \]
\[\text{Term} := \text{<constant symbol> | <individual variable> | f_i (a_0, a_1, ..., a_n) |}
\[\text{To make easier the following definitions, terms can be represented as:}
\[\text{Atomic formula} := X_i^n (t_1, t_2, ..., t_n) \]
\[\text{Wff} := \text{<atomic formula> | (¬ <wff>) | (<wff> → <wff>) | ([<wff> ∧ <wff>] | (<wff> ∨ <wff>) | (¬<wff>) \}
\[\text{Some operations are still being defined, as the Map Analysis Fuzzy Logic Model.} \]
Boolean Logic Modelling involves the combination of several binary maps resulting from the application of conditional operators. These binary maps could be thought as layers of evidence. These layers are combined to support some hypothesis [5]. The AND operator (\( \land \)) is applied to all these layers.

In Binary Evidence Maps, "each map is simply multiplied by its weight factor, summed over all the maps being combined and normalized by the sum of the weights. The result is a value ranging between 0 and 1, which can be classified into intervals appropriate for mapping. At any location, the output score, \( S \), is defined as:

\[
S = \frac{\sum_{i=1}^{n} W_i \cdot \text{class}(MAP_i))}{\sum_{i=1}^{n} W_i},
\]

Where \( W_i \) is the weight of the \( i \)-th map, and \( \text{class}(MAP) \) is either 1 for presence or 0 for absence of the binary condition. The output score is between 0 (implying extremely unfavourable) to 1 (implying highly favourable) [5]. The result is a map showing regions that are ranked according to the score.

In this case, the map classes occurring on each input map are assigned different scores, as well as the maps themselves receiving different weights as before…The average score is then defined by

\[
s = \left( \frac{\sum_{j=1}^{n} s_{ij} \cdot W_j}{\sum_{i=1}^{n} W_i} \right)
\]

where \( s \) is the weighted score for an area object (polygon, pixel), \( W_i \) is the weight for the \( i \)-th input map, and \( s_{ij} \) is the score for the \( j \)-th class of the \( i \)-th map, the value of \( j \) depending on the class actually occurring at the current location [5].

### 6.2 Other Implementation Considerations

The internal representation adopted here, based in quadtrees and Peano relations, allows the storage together of spatial and non-spatial attributes of an object [11]. This characteristic makes possible the use, in the prototype, of whichever Relational (or Object-Relational) Database Management System (DBMS) that exists in the market and talks with Java Database Connectivity (JDBC) [12], [13]. The use of a commercial DBMS adds to the prototype all the advantages and facilities offered by these sophisticated software products. In the prototype we use the Oracle 9i DBMS. The integration among the prototype system and the Oracle 9i is realized by the JDBC classes developed by Sun Microsystems.

During the prototype development, works related to quadtrees, such as OpenMap and Geotools, were considered. But, due to the prototype specific characteristics perhaps we will need to customize some algorithms and classes.

The prototype is being developed using the Java Programming Language [14], [15], [16]. Java has a large library of resources, including the GIS area. Besides that, Java follows the Object-Oriented paradigm and is multi-platform, adding portability to the prototype. Also, Java excels in networking programming, making it possible future expansions of the prototype for Internet/Intranet environments.

### 7. CONCLUSIONS AND FUTURE WORK

The system outlined in this paper, mainly the compiler and graphical user interface, is part of the master thesis, yet in elaboration, of Luis Fernando Bueno, in his master course in Computer Engineering — area of concentration Geomatics at Rio de Janeiro State
University. So, in the next three or four months we expect substantial improve in these modules. We think this is a very simple way of realizing spatial operations with objects, particularly, map analysis and, in this sense, is an integrated approach for spatial object modeling and map analysis. Progressively the prototype could incorporate complex methods and tools not yet supported, due to the open system architecture adopted. The system is also a practical way of working with objects in n-dimensional spaces. Obviously, in this case we'll have the inherent limitations of visualizing n-dimensional objects, when n ≥3. The system could also be used as a prototypical for spatial-temporal geographical information systems. Configurations of spatial objects at different points in time could be thought as a stack of maps. Tools for analyzing the evolution of phenomena in space-time, similar to map analysis tools, could then be developed and applied.

The drawback of the system is the formal way – a 2nd Order Many-Sorted Language, the user needs to interact with the GUI. But, for academic purposes, we think it will be a good tool for formalizing ideas and attaining solutions. We expect also, in the near future, incorporate to the GUI special buttons to define some complex functions and predicates, in an intuitive manner.

8. BIBLIOGRAPHICAL REFERENCES:


