Tạp chí Tin học và Điều khiển học, T. 18, S. 3 (2002), 262–267

SIMPLIFICATION OF THE CONTOURS OF HOUSES

NGO QUOC TAO, PHAM VIET BINH

Abstract. This work is to formally capture the task of map generalisation: how to modify the contents of a map for its changing scale, importance or purpose, while at the same time preserving its geometric and descriptive properties. We use instances of REALM-based spatial data types (formally defined in a companion report [6]) to represent objects on a map and provide several low level operators for the map generalisation: simplification, smoothing, classification, displacement and collapse. We later focus on the simplification of the contour of a house.

Tóm tắt. Công trình này mô tả một số khía cạnh của tổng quát hoá bản đồ bao gồm: cách biến đổi nội dung của bản đồ như thế nào khi thay đổi tỉ lệ, tầm quan trọng và mục đích của bản đồ, đồng thời bảo toàn các tính chất hình học và mô tả bản đồ. Chúng tôi sử dụng các thành phần cơ bản của kiểu dữ liệu không gian REALM (được định nghĩa hình thức trong báo cáo [6]) để biểu diễn các đối tượng trên bản đồ và cung cấp một số toán tử mức thấp cho tổng quát hoá bản đồ như: đơn giản hoá, làm trơn, phân loại, dịch chuyển và cắt dán, sau đó chúng tôi tập trung vào việc đơn giản hoá chu tuyến của ngôi nhà.

1. INTRODUCTION

Map generalisation consists of statistical generalisation and cartographic generalisation. Statistical generalisation is a filtering process whose aim is spatial modelling of attributive information attached to locations. Objective constraints, which are imposed during the process of statistical generalisation, are the preservation of spatial mean, variant and form of the distribution. Cartographic generalisation, the aim of which is visualisation, can affect location accuracy to a great extent.

Map generalisation is a difficult task and depending on the scale, importance and the purpose of the map, requires skilled cartographers working for long periods of time; very few algorithms exist to automate this process. By this paper we want to justify that realm-based structures are useful for developing automatic or semi-automatic software to implement some cartographic operations for map generalisation. And since the problem is hard to solve in the general setting, we gradually restrict out attention to specific simplification of the contour of a house.

We first provide the formal model of a map (Section 2) where each object is an instance of one of REALM-based spatial data-types in [8]: a block, a face or a point. The objects are given attributes relating to what they represent, for instance 'house', 'park', 'street' etc. We next formalise the process of map generalisation (Section 3) . An even larger specialisation is to consider the problem of simplifying the contour of a house (Section 4). The final Section 5 contains conclusions and comments on the directions for future work.

2. MODELLING OF MAPS

A map is a medium of communication that uses labelled graphics to convey spatial relationships among points, lines and areas featured to its viewer [4]. There are many kind of maps depending on their purpose and users, like the maps representing topographical features, hydrography, settlements and roads. On the low level, however, all can be represented as a set of points, lines and areas that are defined by both their location in space (with reference to a system of coordinates and non-spatial attributes ([2]).

In order to formally capture the location part of a map we shall use our earlier work on spatial data types [8] built using the concept and structure of a REALM [5]. The types represent points, lines and regions. All types are closed under the union, intersection and difference and are used as the basic building blocks for representing objects on the map.

A realm R is a set of points and non-intersecting line segments over a finite grid. Given R, point values can be represented as points in R, line values as blocks and region values as sets of edge-disjoint faces in R, each of which may have holes. These are part of the scheme SDT below. Additionally, SDT define auxiliary functions, like *lines_segs* which returns the set of N_segments of an face, is_*regions* expressing the property that a set of faces is a *regions*, is_*lines* for checking whether a set of blocks is a *line*. We also have functions extracting a realm from given points, lines and regions (*points_realm, lines_realm* and *regions_realm*) and functions performing the operations of union, intersection and difference between two type values. The scheme SDT can be written as below.

```
/*————————S D T————————*/
scheme SDT = extend BLOCK with RUNIT with
class type
    lines = {|(bls, R): Block-set × Realm • is_lines(bls, R)|},
    regions = {|(fs, R): Face-set×Realm • is_regions(fs, R)|},
    points = {|(P, R):Grid.N_point-set×Realm • P ⊆ rpoint(R)|}
value is_lines : Block-set×Realm→Bool,
    is_regions : Face-set×Realm→Bool,
    lines_segs : lines→Grid.N_segment-set,
    points_realm : points→Realm,
    lines_realm : lines→Realm,
    regions_realm : regions→Realm,
    p_union, p_intersection, p_difference : points×points −~> points,
    r_union, r_intersection, r_difference : regions×regions −~> regions,
    l_union, l_intersection, l_difference : lines×lines −~> lines
end
```

In order to model a map, all these data-types will be put together. Additionally we need to represent these entities which importance, not size would merit their appearance on the map like for example a hospital. These are usually represented using specialised symbols and will be values of the type *Symbolize*, including a realm, grid, scale and symbol itself. But spatial data-types are able to capture only one aspect of information included in a map. Another aspect is statistical information which with each spatial object would associate an attribute, relating to what an object represents, for instance a 'house', 'park', 'street'... The list, part of the type *Attrib* is open-ended. Then finally an object is a pair of a spatial value and an attribute and a map is a set of objects, all part of the **scheme** MAP below.

```
/*————————M A P————————*/
scheme MAP = extend SDT with
class
type Scale = {|(x, y) : Nat×Nat • x < y |}, Symbol,
    Symbolize :: realm : Realm p : Grid.N_point sc : Scale ss : Symbol,
    Attrib == house | park | street | garden | hospital | _,
    Geo == mk_points(points) | mk_lines(lines) | mk_regions(regions) | mk_Symbolize(Symbolize),
    Object = Geo×Attrib-set,
    Map = {|M : Object-set • is_map(M)|}
value mk_points : points→points,   mk_lines : lines→lines,
    mk_regions : regions→regions, mk_symbolize : Symbolize→Symbolize
    is_map : Object-set→Bool,   realm : Object→Realm,
    realm : Map→Realm
axiom ∀sm : Symbolize • p(sm) ∈ realm(sm),
    ∀M : Object-set •
    is_map(M) ≡ (∀obj, obj1 : Object • obj ∈ M ∧ obj1 ∈ M ⇒ realm(obj) = realm(obj1)),
    ∀(geo, att) : Object •
    realm(geo, att) ≡
        **case** geo **of**
            mk_points(geo) → points_realm(mk_points(geo)),
            mk_lines(geo) →lines_realm(mk_lines(geo)),
            mk_regions(geo) → regions_realm(mk_regions(geo))
        **end,**
    ∀M : Map • realm(M) ≡ **let** obj : Object • obj ∈ M **in** realm(obj) **end**
    **end**
```

3. GENERALISATION OF MAPS

Map generalisation can be considered a process that transforms a map M1 to a map M2. As suggested by Beard (1991) [1], there are four types of constraints for this transformation, these are graphic, structural, application and procedural constraints. Graphic constraints are the limitations of technical equipments and the bounds of human visual ability. Structural constraints are spatial and attribute relations among the objects; they are divided between geometrical, topological and statistical constraints. Application constraints are conditions specific to the purpose of a map.

The constraints are included in the **scheme** GENERALISATION and represented as the predicates *app*(M1,M2) for application, *gr*(M1,M2) for graphical and *str*(M1,M2) for structural constraints between two maps M1 and M2, the latter being a conjuncts of topological (*topo*(M1,M2)), geometrical (*geoconstraint*(M1,M2)) and statistical (*sts*(M1,M2)) constraints; we shall not introduce procedural constraint. We moreover use *r*(eq,M1,M2) to represent limitations of the technical equipment *eq* (*eq* is a value of the abstract type EQ) and a real function Ψ to "measure" the quality of a map (we shall not define this function explicitly).The function max returns the maximum of a set of real numbers.

Then given a map M1 and given technical equipment *eq*, the problem of map generalisation is to find a map M such that *r*(eq,M1,M) and the loss of quality does not exceed the value of θ. Thus M is a 'good enough' approximation of the map M1. These all is part of the **scheme** GENERALISATION.

```
scheme GENERALISATION =
class
type EQ
value  θ: Real, max : Real-set → Real,
    /* graphic constraint*/ gr : EQ × Map × Map → Bool,
    /* relationship between two maps and technical equipments */
    r : EQ × Map × Map → Bool,
    /*topological constraint*/ topo : Map × Map → Bool,
    /* structural constraint*/ str : Map × Map → Bool,
    /* application constraint*/ app : Map × Map −~> Bool,
    /* geometrical constraint*/ geo : Map × Map −~> Bool,
    /* stastistical constraint*/ sts : Map × Map −~> Bool,
    /*the objective function*/ Ψ: Map → Real,
    generalisation : EQ × Map −~> Map,
axiom
    /* Structural constraint */
    ∀ eq: EQ, M, M1, M2 : Map • r(eq, M1, M2) ≡ gr(eq, M1, M2) ∧ str(M1, M2) ∧ app(M1, M2),
    ∀ M1, M2 : Map • str(M1, M2) ≡ geo(M1, M2) ∧ topo(M1, M2) ∧ sts(M1, M2),
    /* The map generalized from the map M1 depending on the capacity of equipment eq */
    ∀ eq : EQ, M1 : Map • generalisation(eq, M1) as M
    post abs (Ψ(M) - max({Ψ(M2) | M2 : Map • r(eq, M1, M2)})) < θ
end
```

The **scheme** OPERATOR provides the abstract definition of the problem only. The solution will typically involve performing a number of low-level operations like classification, simplification, displacement, elimination, selection and they are described below.

+ Classification is a process of grouping objects sharing similar attributes into a new category and perhaps representing them as a new symbol.
+ Simplification is used to reduce the numbers of points representing contours (like for instance of a house).
+ Displacement is an operation that given two objects closer than the value of tolerance.
+ Reduction is a process of removing some objects from the map which scale is reduced.
+ Feature selection is a process of deciding which classes of information, and perhaps which objects within a class, are needed for the purpose of a map [2].
+ Collapse is a process that transforms regions or lines into 'simpler' objects, like lines and points respectively.
+ Typical and Refinement is a process that eliminate objects within a set of similar object while their scatter geometry.

/*——————OPERATORS——————*/
scheme OPERATORS =
class
type classification_result == important |normal| not_important,
    Rule = Object × Scale × Attrib × **Real** × **Real** → **Bool**
value $\theta\_reduction, \theta\_dis1, \theta\_dis2$: **Real**, scale : Scale, User1_Rules, User2_Rules : Rule-set,
    classification : Object → classification_result,
    simplification : feature, exaggeration : Object → Object,
    Collapse, Reduction : Object − ~> Object-set,
    displacement : Object × Object → Object × Object,
    selection, elimination : Object → **Bool**,
    area, ratio : Object → **Real**,
    ...
end

In order to define the operation of simplification we introduce the local function *G2P* returning all vertices of an element of type *Object*. Simplification is used to reduce the numbers of points representing objects in a map:

G2P : Geo → Grid.N_point-set,
simplification : Object → Object
simplification(geo, att) **as** (geo', att) ≡ G2P(geo) ⊇ G2P(geo')

In the **scheme** OPERATORS functions *width, height* and *area* return the width, the height and the area of an object respectively. Moreover *ratio* expresses the ratio between the area of an object and the rectangle embracing this object:

ratio : Object → **Real**
ratio(obj) ≡ area(obj)/(width(obj) * height(obj))

Reduction is a process of removing some objects from the map which scale is reduced.

reduction : Object-set → Object-set,
reduction(objs) **as** obj1s **post card** (objs) > card(obj1s)

Exaggeration is a process of enlarging objects.

exaggeration : Object → Object
exaggeration(obj) **as** obj1 **post** area(obj) < area(obj1)

Given a number of these basic operations, the process of map generalisation will usually invoke them in sequence. It is then important to decide on the kind and the order of invocations.

4. SIMPLIFICATION OF HOUSE CONTOURS

Let us now instantiate the city map to consider the problem of simplifying the contour of a house. In general the contour will be represented by a cycle and we concentrate on four features existing and complicating such contours: short segments, sharp (plain) angles, small areas.

The value cos of the angle between two meeting N_segments.

cos: Grid.N_segment × Grid.N_segment → **Real**

The angle between two N_segments is called 'sharp' if its cos is less than given tolerance SHARP_ANGLE_TOL_VAL.

plainangle : Grid.N_segment × Grid.N_segment →**Bool**,

plainangle(s, s') ≡ cos(s, s') >= PLAIN_ANGLE_TOL_VAL,

Let *h_area*(s,s') represent the area of the polygon with segments s and s' as two opposite edges.

area : Grid.N_segment × Grid.N_segment →**Real**

A triangle forming from two meeting N_segments is called a 'small area' if the area is less than the value of AREA_TOL_VAL.

small_area : Grid.N_segment × Grid.N_segment → **Bool**

small_area(s, s') ≡ Grid.ss_meet(s, s') ∧ area(s, s') < AREA_TOL_VAL,

The simplification should yield the result which 'approximates' the original. Therefore we shall introduce the maximal allowed difference AREA_SUM_TOL_VAL between the number of grid points inside of the original and the transformed contours. Let us first this difference:

```
diff : House × House → Real
diff(h, h') ≡ let(geo, att) = (geo, att), House • h' = (geo', att'), (fs, R) = geo, (fs', R') = geo'
        in  let P_fs ={p|p : Grid.N_point • ∃ f : F.Face • f ∈ fs ∧ pf_areainside(p, f)},
                P_fs' = { p|p : Grid.N_point • ∃ f : F.Face • f ∈ fs ∧ pf_areainside(p, f)},
            in card (P_fs \ P_fs' ∪ P_fs' \ P_fs) end
        end,
```

The operation of simplification can now be defined as below.

```
simplification : House → House
simplification(h) as h'
post let (geo, att) = h, (fs, R) = geo, (geo', att') = h', (fs', R') = geo')
    in ~ (∃ (f0, F0) : F.Face • (f0, F0) ∈ fs' ∧
        (∃ c : C.Cycle, i : Nat • c ∈ {f0}∪ F0 ∧ ( short_segment(c(i)) ∨
    (sharpangle(c(i), c(i+1)) ∨ plainangle (c(i), c(i+1)) ∨ small_area(c(i), c(i+1)))))) ∧
    card (fs') <= card (fs) ∧
    diff(h, h') < AREA_TOL_VAL
    end
```

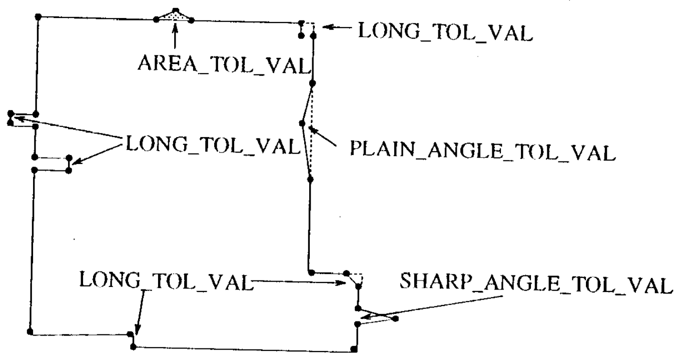

Figure 1. The small formes and the tolerance values

The specification of simplification operation is used to verify some simplification algorithm of contour of house. Figure 1 shows that the algorithm of Damour [3] for simplifying the contours of house increases the numbers of the house.

5. CONCLUSION

The paper provides the formal representation of the map generalisation problem and provides the REALM-based model for maps and abstract definitions for the basic operations on maps. Particular attention is directed towards the problem of simplifying the contour of house.

In future we plan to give some more concrete formal descriptions of some map generalisation operators: classification, simplification/smoothing, selection/elimination and collapse, all considered for the class of city maps and its typical objects like streets, parks, gardens and hospitals. We moreover want to study the automated generalisation of lines, applying this to simplify streets.

REFERENCES

[1] Beard M.K., *Map generalization: Making rules for knowledge representation*, Longman Science & Technical Pub., 90 Tottenham Court Road, London W1P9HE, 1 edition, 1991.

[2] Buttenfield B.P., McMaster R.B., and Freeman H., editors, *Map generalization: Making rules for knowledge representation*, Longman Science & Technical Pub., 90 Tottenham Court Road, London W1P9HE, 1 edition, 1991.

[3] Damour S., Generalisation du bati : la simplification des contours, Rapport, COGIT Laboratoire Service - Institute Geographique National (IGN), 2, Avenue Pasteuer - BP 68 F-94160 Saint-Mande France, 30 july 1991.

[4] Freeman H., *Geographic Information Systems*, Vol. 1, Longman Scientific & Technical, London, 1991, 457–475.

[5] Gusting R.H. and Schneider M., Realm: A Foundation for Spatial Data Types in Database Systems, In David Abel and Beng Chin Ooi, editors, *Advances in Spatial Database- Third International Symposium, SSD'93 Singapore, June 23-25, 1993 Proceedings*, Berlin, June 1993.

[6] Largrange J.P., Ruas A., and Bender L., Survey on Generalization, Unpublished, COGIT Laboratoire Service - Institute Geographique National (IGN), 2, Avenue Pasteuer - BP 68 F-94160 Saint-Mande France, 1993.

[7] Muller J.C., *Geographic Information Systems*, Longman Scientific & Technical Pub., London, 1991 (457–475.

[8] Ngo Quoc Tao, A Formal Specification of Realm - Spatial Data Type, *Research Report 71, UNU/IIST*, P.O.Box 3058, Macau, April 1996.

Received April 26, 2002

Pham Viet Binh - Thai Nguyen University
Ngo Quoc Tao - The Institute of Information Technology