About Invalid, Valid and Clean Polygons

Peter van Oosterom, Wilko Quak and Theo Tijssen

Delft University of Technology, OTB, section GIS technology, Jaffalaan 9, 2628 BX Delft, The Netherlands.

Abstract

Spatial models are often based on polygons both in 2D and 3D. Many Geo-ICT products support spatial data types, such as the polygon, based on the OpenGIS 'Simple Features Specification'. OpenGIS and ISO have agreed to harmonize their specifications and standards. In this paper we discuss the relevant aspects related to polygons in these standards and compare several implementations. A quite exhaustive set of test polygons (with holes) has been developed. The test results reveal significant differences in the implementations, which causes interoperability problems. Part of these differences can be explained by different interpretations (definitions) of the OpenGIS and ISO standards (do not have an equal polygon definition). Another part of these differences is due to typical implementation issues, such as alternative methods for handling tolerances. Based on these experiences we propose an unambiguous definition for polygons, which makes polygons again the stable foundation it is supposed to be in spatial modelling and analysis. Valid polygons are well defined, but as they may still cause problems during data transfer, also the concept of (valid) clean polygons is defined.

1 Introduction

Within our Geo-Database Management Centre (GDMC), we investigate different Geo-ICT products, such as Geo-DBMSs, GIS packages and 'geo' middleware solutions. During our tests and benchmarks, we noticed subtle, but fundamental differences in the way polygons are treated (even in the 2D situation and using only straight lines). The consequences can be quite unpleasant. For example, a different number of objects are selected when the same query is executed on the same data set in different environments. Another consequence is that data may be lost when transferring it from one system to another, as polygons valid in one environment may not be accepted in the other environment.

It all seems so simple, everyone working with geo-information knows what a polygon is: an area bounded by straight-line segments (and possibly having some holes). A dictionary definition of a polygon: a figure, (usually a plane, rectilinear figure), having many, i.e. (usually) more than four, angles (and sides) (Oxford 1973). A polygon is the foundation geometric data type of many spatial data models, such as used for topographic data, cadastral data, soil data, to name just a few. So, why have the main Geo-ICT vendors not been able to implement the same polygons? The answer is that in reality the situation is not as simple as it may seem at first sight. The two main difficulties, which potentially cause differences between the systems, are:

1. Is the outer boundary allowed to interact with itself and possibly also with the inner boundaries and if so, under what conditions?
2. The computer is a finite digital machine and therefore coordinates may sometimes differ a little from the (real) mathematical value. Therefore tolerance values (epsilons) are needed when validating polygons.

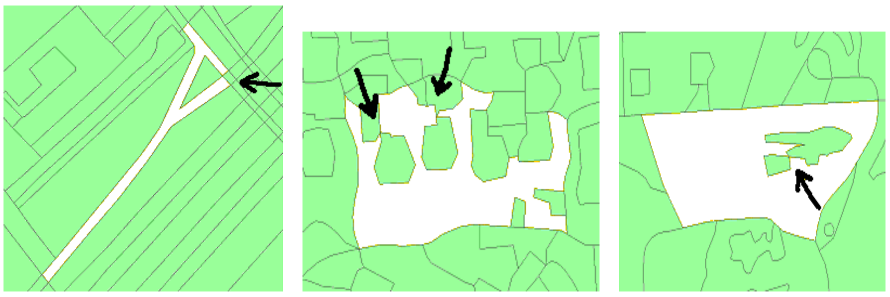

Fig. 1. Real world examples: topographic data (left: outer ring touches itself in one point, middle: two inner rings (holes) both touch the outer ring, right: two inner rings touch each other)

The interaction between the outer and possibly the inner boundaries of a single polygon is related to the topological analysis of the situation. This is an abstract issue, without implementation difficulties, such as tolerance values. So, one might expect that the main 'geometry' standards of Open-GIS and ISO will provide a clear answer to this. A basic concept is that of a straight-line segment, which is defined by its begin and end point. Polygon input could be specified as an unordered, unstructured set of straight-line segments. The following issues have to be addressed before it can be decided whether the set represents a valid or invalid polygon (also have a look a the figures in section 3):

1. Which line segments, and in which order, are connected to each other?
2. Is there one or are there more than one connected sets of straight-line segments?
3. Are all connected sets of straight-line segments closed, that is, do they form boundary rings and is every node (vertex) in the ring associated with at least 2 segments?
4. In case a node is used in 4, 6, 8, etc. segments of one ring, is the ring then disconnected in respectively 2, 3, 4, etc. 'separate' rings (both choices may be considered 'valid', anyhow the situation can happen in reality irrespective of how this should be modeled, see fig. 1)?
5. Are there any crossing segments (this would not be allowed for a valid polygon)?
6. Is there one ring (the outer ring), which encloses an area that 'contains' all other (inner) rings?
7. Are there no nested inner rings (would result in disconnected areas)?
8. Are there any touching rings? This is related to question 4, but another situation occurs when one ring touches with one of its nodes another ring in the interior of a straight-line segment.
9. Are the rings, after construction from the line segments, properly oriented, that is counter clockwise for the outer boundary and clockwise for the inner boundaries? (defining a normal vector for the area that points upward as a usual convection inherited from the computer graphics world: only areas with a normal vector in the direction of the viewer are visible) Note that this means that the polygon area will always be on the left-hand side of the polygon boundaries.

Some of questions may be combined in one test in an actual implementation in order to decide if the polygon is valid. In case a polygon is invalid, it may be completely rejected (with an error message) or it may be 'accepted' (with a warning) by the system, but then operations on such a polygon may often not be guaranteed to be working correctly. In this paper it is assumed that during data transfer between different systems enough characters or bytes are used in case of respectively ACSII (such as GML of OpenGIS and ISO TC211) or binary data formats in order to avoid unwanted change of coordinates (due to rounding/conversion) and that the sending and receiving system have similar capabilities for representing coordinates; e.g. integers (4 bytes), floating point numbers (4, 8, or 16 bytes) (IEEE 1985). In reality this can also be a non-trivial issue in which errors might be introduced. A worst case scenario would be transferring the same data several times between two systems (without editing) and every time the coordinates drift further away due to rounding/conversions. By using enough characters (bytes) this should be avoided.

In reality, polygons are not specified as a set of unconnected and unordered straight-line segments. One reason for this it that every coordinate would then at least be specified twice and this would be quite redundant with all associated problems, such as possible errors and increased storage requirements. Therefore, most systems require the user to specify the polygon as a set of ordered and oriented boundary rings (so every coordinate is stated only once). The advantage is also that many of the tasks listed above are already solved through the syntax of the polygon. But the user can still make errors, e.g. switch the outer and inner boundary, or specify rings with erroneous orientation. So, in order to be sure that the polygon is valid, most things have to be checked anyhow.

2 Polygon definitions

In this section we review a number of polygon definitions. First, we have a look at the definition of a (simple) polygon as used within computational geometry. Then the ISO and the OpenGIS polygon definitions are discussed. Then, we present our definition, which tries to fill the blank spots of the mentioned definitions and in case of inconsistencies between the standards make a decision based on a well defined (and straight forward) set of rules. Finally, the concept of clean (and robust) polygons is introduced.

2.1 Computational geometry

From the computational text book of Preparata and Shamos (1985, p.18): *'a polygon is defined by a finite set of segments such that every segment extreme is shared by exactly two edges and no subset of edges has the same property.'* This excludes situations with dangling segments, but also excludes two disjoint regions (could be called a multi-polygon), polygon with a hole, or a polygon in which the boundary touches itself in one point (extreme is shared by 4, 6, 8, … edges). However, it does not exclude a self-intersecting polygon, that is, two edges which intersect. Therefore also the following definition is given: *'A polygon is simple if there is no pair of nonconsecutive edges sharing a point. A simple polygon partitions the plane into two disjoint regions, the interior (bounded) and the exterior (unbounded).'* Besides the self-intersection polygons, this also disallows polygons with (partial) overlapping edges. Finally the following interesting remark is made: *'in common parlance, the term polygon is frequently used to denote the union of the boundary and the interior.'* This is certainly true in the GIS context, which implies that actually the simple polygon definition is intended as otherwise the interior would not be defined. One drawback of this definition is that is disallows polygons with holes, which are quite frequent in the GIS context.

2.2 ISO definition

The ISO standard 19107 'Geographic information — Spatial schema' (ISO 2003) has the following polygon definition: *'A GM_Polygon is a surface patch that is defined by a set of boundary curves (most likely GM_CurveSegments) and an underlying surface to which these curves adhere. The default is that the curves are coplanar and the polygon uses planar interpolation in its interior.'* It then continues with describing the two important attributes, the exterior and the interior: *'The attribute "exterior" describes the "largest boundary" of the surface patch. The GM_GenericCurves that constitute the exterior and interior boundaries of this GM_Polygon shall be oriented in a manner consistent with the upNormal of the this.spanningSurface.'* and *'The attribute "interior" describes all but the exterior boundary of the surface patch.'* Note that in this context the words exterior and interior refer to the rings defining respectively the outer and inner boundaries of a polygon (and not referring to the exterior area and interior area of the polygon with holes).

It is a bit dangerous to quote from the ISO standard without the full context (and therefore exact meaning of primitives such as GM_CurveSegments and GM_GenericCurves). The GM_Polygon is a specialization of the more generic GM_SurfacePatch, which has the following ISO definition: *'GM_SurfacePatch defines a homogeneous portion of a GM_Surface. The multiplicity of the association "Segmentation" specifies that each GM_SurfacePatch shall be in one and only one GM_Surface.'* The ISO definition for the GM_Surface is: *'GM_Surface, a subclass of GM_Primitive, is the basis for 2-dimensional geometry. Unorientable surfaces such as the Möbius band are not allowed. The orientation of a surface chooses an "up" direction through the choice of the upward normal, which, if the surface is not a cycle, is the side of the surface from which the exterior boundary appears counterclockwise. Reversal of the surface orientation reverses the curve orientation of each boundary component, and interchanges the conceptual "up" and "down" direction of the surface. If the surface is the boundary of a solid, the "up" direction is outward. For closed surfaces, which have no boundary, the up direction is* *that of the surface patches, which must be consistent with one another. Its included GM_SurfacePatches describe the interior structure of a GM_Surface.'*

So, this is not the simple definition of a polygon one might aspect. Further, it is not directly obvious if the outer boundary is allowed to touch itself or if it is allowed to touch the inner boundaries and if so, under what conditions this would be allowed. One thing is very clear: there is just one outer boundary and there can be zero or more inner boundaries. This means that a 'polygon' with two outer boundaries, defining potentially disconnected areas, is certainly invalid. Also the ISO standard is very explicit about the orientation of the outer and inner boundaries (in 2D looking from above: counterclockwise and clockwise for respectively the outer and inner boundaries).

2.3 OpenGIS definition

The ISO definition of a polygon is at the abstract (mathematical) level and part of the whole complex of related geometry definition. The definition has to be translated to the implementation level and this is what is done by the OpenGIS Simple Feature Specification (SFS) for SQL (OGC 1999). The OpenGIS definition is based on the ISO definition, so it can be expected that there will (hopefully) be some resemblance: *'A Polygon is a planar Surface, defined by 1 exterior boundary and 0 or more interior boundaries. Each interior boundary defines a hole in the Polygon. The assertions for polygons (the rules that define valid polygons) are:*

1. *Polygons are topologically closed.*
2. *The boundary of a Polygon consists of a set of LinearRings that make up its exterior and interior boundaries.*
3. *No two rings in the boundary cross, the rings in the boundary of a Polygon may intersect at a Point but only as a tangent:*
   $\forall\ P \in \mathit{Polygon},\ \forall c1, c2 \in P.\mathit{Boundary}(),\ c1 \neq c2,$
   $\forall\ p, q \in \mathit{Point},\ p, q \in c1,\ p \neq q,\ [\,p \in c2 \Rightarrow q \notin c2]$
4. *A Polygon may not have cut lines, spikes or punctures:*
   $\forall\ P \in \mathit{Polygon},\ P = \mathit{Closure}(\mathit{Interior}(P))$
5. *The Interior of every Polygon is a connected point set.*
6. *The Exterior of a Polygon with 1 or more holes is not connected. Each hole defines a connected component of the Exterior.*

In the above assertions, Interior, Closure and Exterior have the standard topological definitions. The combination of 1 and 3 make a Polygon a Regular Closed point set.'

Similar to the ISO standard, in the OpenGIS SFS specification, a polygon is also a specialization of the more generic surface type, which can exits in 3D space. However, the only instantiable subclass of Surface defined in the OpenGIS SFS specification, Polygon, is a simple Surface that is planar. As might be expected from an implementation specification a number of things become clearer. According to condition 3: rings may touch each other in at most one point. Further condition 5 makes clear that the interior of a polygon must be a connected set (and a configuration of inner rings, which somehow subdivides the interior of a polygon into disconnected parts, is not allowed). Finally, an interesting point is raised in condition 4: cut lines or spikes are not allowed. All-fine from a mathematical point of view, but when is a 'sharp part' of the boundary considered a spike. Must the interior angle at that point be exactly 0, or is some kind of tolerance involved. The same is true for testing if a ring touches itself or if two rings touch each other (in a node-node situation or a node-segment situation). Note that the OpenGIS does not say anything concerning the orientation of the polygon rings.

2.4 An enhanced 'polygon with holes' definition

Our definition of a *valid polygon with holes*: 'A polygon is defined by straight-line segments, all organized in rings, representing at least one outer (oriented counterclockwise) and zero or more inner boundaries (oriented clockwise, also see sections 2.1-2.3 for used concepts). This implies that all nodes are at least connected to two line segments and no dangling line segments are allowed. Rings are not allowed to cross, but it is allowed that rings touch (or even partially) overlap themselves or each other, as long as *any point inside or on the boundary of the polygon can be reached through the interior of the polygon from any other point inside the polygon*, that is, it defines one connected area. As indicated above, some conditions (e.g. 'ring touches other ring') require a tolerance value in their evaluation and therefore this is the last part of the definition.'

One could consider not using a tolerance value and only look at exact values of the coordinates and the straight-lines defined by them. Imagine a situation in which a point of an inner ring is very close to the outer ring; see for example cases 4, 31 and 32 in the figure of section 3. The situation in reality may have been that the point is supposed to be located on the ring (case 4). However, due to the finite number of available digits in a computer, it may not be possible to represent that exact location (Goldberg 1991, Güting 1993), but a close location is chosen (cases 31 and 32). It is arbitrary if this point would be on the one or the other side of the ring. Not considering tolerances would either mean that this situation would be classified as crossing rings (not allowed) or two disjoint rings (is not the case, as they are supposed to touch each other). Anyway, the polygon (ring and validity) situation is not correctly assessed. This is one of the reasons why many systems use some kind of tolerance value. The problem is how to specify the manner the tolerance value is applied when validating the polygon (this part is missing in our definition above). Another example, which illustrates this problem, is case 30 (see figure in section 3): one option for tolerance processing could be to remove the 'internal spike' and the result would be a valid polygon. However, an alternative approach may be to widen the gap between the toe end nodes and in this situation the result is an invalid polygon as the 'internal spike' intersects one of the other edges. It may be difficult to formalize unambiguous epsilon processing rules as a part of the validation process.

Another strange aspect of our definition of valid polygons is that a 'spike to the outside' (case 11 in figure section 3) results in an invalid polygon as it is not possible from a point in the middle of this spike to reach all other points of the polygon via the interior. While at the same time a 'spike to the inside' (case 12 in figure section 3) is considered a valid polygon as it is possible to reach from any point of the polygon (also from the middle of the spike) any other point via the interior of the polygon. Something similar as with 'spikes' occurs with 'bridges': while internal bridges are valid (cases 7 and 8), the external bridges (case 15 and 16) are invalid. Both in the situation of spikes and bridges, the difference between internal and external could be considered 'asymmetrical'.

2.5 Valid and clean polygons

The validation process (according to our definition of valid polygons as described above) would become much simpler is it can be assumed that *no point lies within epsilon tolerance of any other point or edge* (which it does not define itself), which will be called a (valid) *clean polygon*. In cases 4 (31 and 32) this implies that the segment on which the point of the other ring is supposed to be located, should be split into two parts, with the point as best possible representation within the computer. By enforcing this way of modelling, the polygon validity assessment may be executed without tolerances. Another advantage of clean polygons as described above is that they will not have any spikes (not to the inside and not to the outside) as the two end nodes of the spike lay too close together (or are even equal). Similarly, the internal and external bridges should be removed. Further, repeated points are removed from the representation. Before transferring polygon data the sending system therefore first do the epsilon tolerance processing. After that, the sender can be sure that the receiving system will correctly receive the polygons (assuming that coordinates are not changed more that epsilon during transfer).

The largest distance of moving a coordinate, while the result is still a valid polygon, is called the *robustness* of the polygon representation of Thompson (2003). In case 4 without an additional node, the robustness would be equal to 0 as infinitely small change (to the outside) of the node of the inner ring on the edge of the outer ring would make this polygon invalid (not considering epsilon tolerance). However, adding an explicit shared node in both inner and outer ring as result of the epsilon tolerance processing increases the robustness of this representation (of the 'same' polygon) to at least the value of epsilon. In fact the robustness is even larger as it is possible to change every (shared) node by more than epsilon (the size of epsilon can be observed from the open circle in the drawing of cases 31 and 32). The robustness of a polygon can be computed finding the smallest distance between a node and an edge (not defined by the node). The smallest distance can be either reached somewhere in the middle of near one of the end points of the involved edge. A brute force algorithm would require O(n^2), while a smarter (computational geometry) algorithm could probably compute this in O(n log n), where n is the number of nodes (or edges) in the polygon. The concept of robustness has some resemblance with the 'indiscernibility' relation between two representations as introduced by Worboys (1998).

3 Testing Geo-ICT systems

In this section we first specify a list of representative test polygons. This list is supported to be exhaustive for all possible valid and invalid type of polygons. Next we use this test set in combination with four different geo-DBMSs and compare the outcome to the OpenGIS, ISO and our own definition of valid polygons.

3.1 Polygon examples

Figure 2 shows an overview of our test polygons. In these images, small filled circles represent nodes. Sometimes, two of these circles are drawn very close to each other; this actually means that the nodes are exactly the same. The same is true for two very close lines, which actually mean (partly) overlapping segments. Some figures contain empty circles, which indicate tolerance values (the assumed tolerance value is 4000 related to the coordinates used in our test polygons). In case such a situation occurs, the system may decide to 'correct' the polygon, within the tolerance value distances, resulting in a new polygon representation. The resulting new representation may be valid or invalid, depending on the configuration.

Note that all polygon outer rings are defined counterclockwise and all inner rings are defined clockwise. More test variants can be imagined when reversing the orientation (only done for test 1). Further, rings touching itself can be modelled as one ring or modelled as separate rings. The separate ring option is chosen, with the exception of example 4, where both cases are tested: 4a, the 'separate rings' option (without explicit node were rings touch), 4b, the 'single ring' option (self touching). Even a third variant would be possible: two 'separate rings' with explicit nodes were these rings touch (not tested). Also in this case more test variants can be imagined. In addition to our presented set of test cases, it may be possible to think of other test cases. It is important to collect these test cases in order to evaluate the completeness (and correctness) of the standards related. To name just a few additional, untested, cases (but many more will exist):

- polygons with inner and outer ring switched, (first inner ring specified, then outer ring), but both with the proper orientation
- same as above, but now also the orientation (clockwise/ counterclockwise) reversed.
- two exactly the same points on a straight line (similar to case 26, but now with repeated points)
- same as above, but now the two points are repeated on a true 'corner' of the polygon
- line segment of inner ring is within tolerance of a line segment of the outer ring (but on the inside), similar to case 9 but with tolerance value same as above but now the line segment is on the outside
- two outer rings, with partly overlapping edges or touching in point

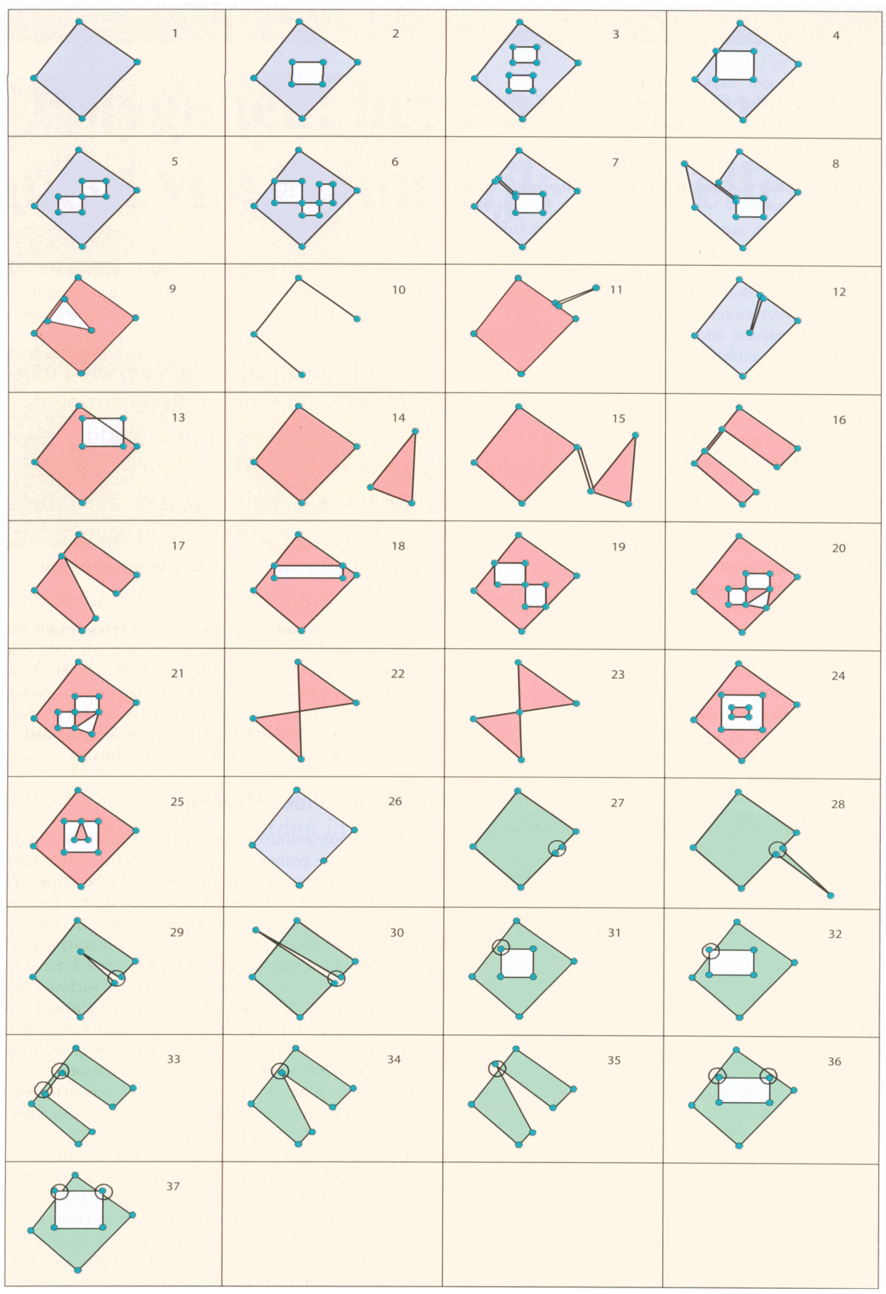

Fig. 2. Overview of the polygons used in the test

Table 1. Results of validating the polygons, no code means polygon is considered valid (BS=boundary selfintersects, CR=crossing rings, EN=edge not connected to interior, FI=floating inner ring, NA=no area, NC=not closed, NH=not one homogenous portion, NO=not orientable, NS=no surface, R*n*=rule *n* (*n*=1,3,4,5), RC=ring crosses ring, RO=rings overlap, RT=rings touch, SR=self crossing ring, TE=two exterior rings, TS=two separate areas, WO=wrong orientation)

id	Oracle	Informix	PostGIS	ArcSDE	OGC-SFS	ISO 19107	we
1a							
1b	WO					WO	WO
2							
3							
4a		BS		RC			
4b	BS				R3		
5							
6		BS		RC			
7	BS			RC			
8	BS	BS		RC			
9	RO	BS		RC	R3	?	EN
10	NC	NC	NC		R1	NS	NA
11	BS			RC	R4	?	EN
12	BS			RC	R4		
13	RT	BS		FI	R3	?	CR
14a	WO				R5	WO	TS
14b					R5	TE	TS
15	BS			RC	R5	NH	TS
16	BS	BS		RC	R5	NH	TS
17	BS	BS		RC	R5	NH	TS
18	RT	BS		RC	R5	NH	TS
19	BS	BS		RC	R5	NH	TS
20		BS		RC	R5	NH	TS
21					R5	NH	TS
22	BS	BS		RC	R3	NO	SR
23	BS				R3	NH	TS
24	WO				R5	TE	TS
25	BS				R5	NH	TS
26							
27							
28					?	?	
29					?	?	
30	BS	BS		RC	?	?	
31							
32	RT	BS		RC	?	?	
33					?	?	TS
34					?	?	TS
35	BS	BS		RC	?	?	TS
36					?	?	TS
37	RT	BS		RC	?	?	TS

3.2 System tests

We tested a test set of about 40 polygons in different spatial databases: Oracle (2001), Informix (2000), PostGIS (Ramsey 2001, PostgreSQL 2001), and ArcSDE binary (ESRI). Of these implementations Informix and PostGIS use the OpenGIS specification for polygons. Oracle Spatial defines a polygon as: '*Polygons are composed of connected line strings that form a closed ring and the area of the polygon is implied.'* The OpenGIS Well Known Text (WKT) format was used when trying to insert the records in the different systems, with the exception of ArcSDE (see below). Because Oracle does not support this format we integrated the Java Topology Suite (1.3) into the Oracle server to implement the conversion function. Below a single example of the exact inserts statement for three (of the four) systems of a correct polygon with one hole (case 2):

Oracle example:

```
insert into test_polygon values ('2', GeomFromText( 'polygon(
 (33300 19200, 19200 30000, 8300 15000, 20000 4200, 33300 19200),
 (25000 13300, 17500 13300, 17500 19200, 25000 19200, 25000 13300))'));
```

Informix example:

```
insert into test_polygon values ('2', ST_PolyFromText('polygon(
 (33300 19200, 19200 30000, 8300 15000, 20000 4200, 33300 19200),
 (25000 13300, 17500 13300, 17500 19200, 25000 19200, 25000 13300))',
 128992));
```

PostGIS/PostgreSQL:

```
insert into test_polygon values ('2', GeometryFromText('POLYGON(
 (33300 19200, 19200 30000, 8300 15000, 20000 4200, 33300 19200),
 (25000 13300, 17500 13300, 17500 19200, 25000 19200, 25000 13300))',
  -1));
```

Oracle has a separate validation function in which a parameter for the tolerance can be specified (example below shows our tolerance value of 4000):

```
select id, sdo_geom.validate_geometry_with_context(geom,4000) as valid
  from test_polygon
  where sdo_geom.validate_geometry_with_context(geom,4000) <> 'TRUE';
```

It was not possible to load the WKT format into ArcSDE (without writing a program using ESRI's ArcSDE API). As we did want to be sure to input the same polygon definitions with standard tools of the vendor we used the following procedure to load the data into ArcSDE:

1. WKT was converted by hand to 'mif' ('mid') format: in this manner exactly the same coordinates, ordering and rings were specified (including the repetition of first and last point)

2. ArcTools 8.3 was used to convert the mif/mid format to ESRI shape files, a binary format. This conversion has as a side effect some coordinate manipulation: e.g. outer rings are now always ordered clockwise, a repeated last point is omitted, rings defining a region are per definition closed. This makes it impossible to test case 10.
3. Finally the polygons are loaded into ArcSDE binary with the following ArcSDE command:
```
create -l testpoly,geom. -f testpoly.shp -e a+ -k SDEBINARY \
     -9 10000 -a all -u username
```

The table gives an overview of the different responses by the four systems. This table also contains the expected results according to the ISO and OpenGIS definitions and our own definition. The result of inserting the test cases in the four different DBMSs leads to the following general observations. Oracle Spatial (version 9.2.0.3.0) provides a tolerance parameter that is used with many of the operations. In this test a tolerance of 4000 was used. Experiments with different tolerance values yielded the same results. If Informix (Spatial DataBlade Release 8.11.UC1 (Build 225)) finds a polygon with erroneous orientation, it reverses the polygon internally without warning. PostGIS 0.6.2 (on PostgreSQL 7.1.3) only supports the GeomFromText (and not PolyFromText) and the geometries cannot be validated.

4 Conclusion

As noticed in our experience when benchmarking and testing geo-ICT products, the consistent use of a polygon definition is not yet a reality. This is both true for the standards (specifications) and the implementation in products. Both based on the ISO 19107 standard and OpenGIS SFS for SQL implementation specification it may sometimes be very difficult or impossible to determine whether a polygon is valid or not. Also according to our evaluation of the set with test cases, the results of (in)valid polygons are not always harmonized between ISO and OpenGIS. Further, both ISO and OpenGIS definitions do not cover the important aspect (when implementing polygons) of tolerance value. Therefore our own improved definition of a polygon (with holes) was given. This was refined by the definition of a (valid) clean polygon, which is suitable for data transfer (and easy validation).

A part of the polygon validation may already be embedded in the syntax of the 'polygon input' (string) and certain validation tasks are implicit (e.g. the system does not have to assemble the ring from the individual straight line segments as the rings are specified). One could wonder if the orientation of the rings in a polygon should be a strict requirement as the intended polygon area is clear. In case in the syntax the outer polygon ring would not be determined by the ordering of the rings (outer ring first), but purely by the orientation of the rings (outer ring could be any one in the list of rings), then proper orientation is useful as this can be used to detect inner and outer rings. But even this it is not strictly needed as one could also determine via the geometric configuration (computing) which ring should be considered the outer ring.

Besides the theory, our tests with four different Geo-DBMSs, Oracle, Informix, PostGIS, and ArcSDE binary and one geo middleware product (LaserScan Radius Topology, not reported here), revealed that also in practice significant differences in polygon validation do exist. It needs no further explanation that this will cause serious problems during data transfer, including loss of data. We urge standardization organizations and Geo-ICT vendors to address this problem and consider our proposed definition.

Until now, only the validation of (input) polygons is discussed, but what happens with these (in)valid polygons during operations; e.g. intersection of two valid polygons may result in disconnected areas (that is not a valid polygon). How is the area or perimeter of a polygon computed in case it is specified in long, lat (on the curved surface such as an sphere, ellipsoid or geoid), What will be the resulting units and how does the tolerance value influence this result?

In this paper only simple polygons with holes on a flat surface were discussed. However as already indicated in the previous paragraph (curved surfaces), more complex situations can occur in the world of geo-ICT products (and standards):

- Multi-polygons (that is, two or more outer boundaries which are not connected to each other)
- Also include non linear edges in boundary (e.g. circular arcs)
- In 3D space, but limited to flat surface
- In 3D space, but limited to polyhedral surfaces (piecewise flat)
- In 3D, non flat surfaces, especially an Earth ellipsoid (or geoid)

For all these situations unambiguous and complete definitions, including the tolerance aspect, must be available. Test cases should be defined and subsequently the products should be valuated with these test cases. And after finishing with polygons, we should continue with polyhedrons (Arens et al. 2003).

Acknowledgements

We would like to thank the Topographic Service and the Dutch Cadastre for providing us with the test data sets (since January 2004, the Topographic Service is part of the Cadastre). We are further grateful to the vendors and developers of the Geo-ICT products mentioned in this paper (Oracle, Informix, PostgreSQL/PostGIS, ArcSDE), for making their products available for our research. Finally, we would like to thank the anonymous reviewers of this paper for the constructive remarks.

References

Arens C, Stoter JE, van Oosterom PJM (2003) Modelling 3D spatial objects in a Geo-DBMS using a 3D primitive, Proceedings 6th AGILE, Lyon, France.

Goldberg D (1991) What Every Computer Scientist Should Know About Floating-Point Arithmetic, ACM Computing Surveys, Vol. 23: 5-48.

Güting R and Schneider (1993) Realms: A foundation for spatial data types in database systems. In D. J. Abel and B. C. Ooi, editors, Proceedings of the 3rd International Symposium on Large Spatial Databases (SSD), volume 692 of Lecture Notes in Computer Science, pages 14-35. Springer-Verlag.

IEEE (1985) American National Standard -- IEEE Standard for Binary Floating Point Arithmetic. ANSI/IEEE 754-1985 (New York: American National Standards Institute, Inc.).

Informix (2000) Informix Spatial DataBlade Module User's Guide. December 2000. Part no. 000-6868.

ISO (2003) ISO/TC 211/WG 2, ISO/CD 19107, Geographic information — Spatial schema, 2003.

OGC (1999) Open GIS Consortium, Inc., OpenGIS Simple Features Specification For SQL, Revision 1.1, OpenGIS Project Document 99-049, 5 May 1999.

Oracle (2001) Oracle Spatial User's Guide and Reference. Oracle Corporation, Redwood City, CA, USA, June 2001. Release 9.0.1 Part No. A8805-01.

Oxford (1973) The Shorter Oxford English dictionary.

PostgreSQL (2001) The PostgreSQL Global Development Group. PostgreSQL 7.1.3 Documentation.

Preparata FP and Shamos MI (1985) Computational Geometry, an Introduction. Springer-Verlag, New York Berlin Heidelberg Tokyo.

Ramsey P (2001) PostGIS Manual (version 0.6.2). Refractions Research Inc.

Thompson R (2003) PhD research proposal 'Towards a Rigorous Logic for Spatial Data Representation'. Department of Geographical Sciences and Planning, The University of Queensland, Australia, November 2003.

Worboys MF (1998) Some Algebraic and Logical Foundations for Spatial Imprecision, in Goodchild M. and Jeansoulin, R (ed), Data Quality in Geographic Information: from error to uncertainty, Hermes.