Formalization of Spatial Constraints

Stefan Werder
Institute of Cartography and Geoinformatics, Leibniz Universität Hannover
Stefan.Werder@ikg.uni-hannover.de

ABSTRACT

Constraints are inherent in most spatial datasets. However, they are often not defined explicitly and therefore cannot be enforced. This leads to the need for an explicit formalization of spatial constraints. In order to identify the diverse formalization requirements, previous research on constraint classification and usage in both data integration and generalization is analyzed. Also different approaches for modeling and enforcing constraints are presented. As a result from the analysis, a flexible and expressive constraint formalization based on the Object Constraint Language (OCL) is proposed. It is then shown how the so-called GeoOCL is able to satisfy most of the identified formalization requirements.

1 INTRODUCTION

Ensuring consistency of spatial data is a crucial and often complex task. Besides technical issues, consistency can be checked if the underlying constraints are properly modeled and enforced. The term constraint thereby includes alpha-numerical as well as spatial definitions. In order to allow for automated checking, constraints have to be modeled explicitly.

Research on classification and usage of constraints as well as modeling and enforcing constraints has been carried out in several disciplines. The most important are data integration and generalization. However, little research exists that incorporates both disciplines, although they are tightly related.

In generalization, constraints specify in most cases the requirements that the target product has to meet, e.g. concerning the legibility of a produced map. In data integration, being in the focus of our research, constraints are mainly used for checking data integrity. Concerning data integrity, constraints can be enforced on existing datasets but also when datasets of different theme, acquisition time, or quality have to be conflated. An example for a spatial constraint being used in both disciplines is the enforcement of minimum length of line segments. In generalization this constraint ensures legibility, whereas in data integrity the correct modeling respectively acquisition is enforced, i.e. street line segments being longer than 0.2 m.

Normally constraints can be specified for a dataset in general. Nevertheless, they can be also specific based on the target application. In some cases they originate directly from the way data acquisition is organized, e.g. for rectangular houses often only two sides are measured. Houses are then modeled with four right angles leading to parallel sides of equal length.

In our project we use constraints for checking data integrity during a data integration workflow. If two datasets have to be conflated, first the constraints on each dataset are evaluated. Then they are combined and the constraints concerning both datasets are evaluated. Because the data integrity process shall be used as part of a Spatial Data Infrastructure (SDI), it is implemented based on the Web Processing Service (WPS) standard [17] established by the Open Geospatial Consortium (OGC). Implementing for an SDI implies the usage of other common standards too, which is shown throughout the paper.

The rest of the paper is structured as follows. In Sect. 2 the requirements for constraint formalization are collected from the classification and usage of constraints in both data integration and generalization research. In Sect. 3 the diverse approaches for modeling and enforcing constraints are presented. The synthesis of the requirements and approaches finally leads to the proposal of a flexible and expressive constraint formalization language called GeoOCL in Sect. 4.

2 CLASSIFICATION AND USAGE OF CONSTRAINTS

For the creation of a flexible and expressive constraint formalization, the requirements of several scenarios, applications, and domains have to be considered. Therefore the general requirements concerning constraints that have been identified and modeled in a selection of data models and diverse research projects are shown in this section. In Sect. 4.4 we then check to which extent our proposed solution adheres to the given requirements. We present research on constraints from both the data integration and the generalization community, because they represent two sides of the same coin.

Building a single taxonomy covering all diverse classification schemata presented in the following, seems from our point of view an impracticable, if not impossible, task. Therefore we didn't address the issue to which respective constraint subclass of an author individual constraints such as perpendicularity are assigned to. In order to compare several works at that detailed level we propose to label individual constraints with adequate tags.

2.1 Constraints in data integration

Integrity constraints are used by Cockcroft [5] for improving spatial data quality. Constraints are enforced upon insertion of features into a data set. The concept of severity is used by Cockcroft to produce different types of warnings and for consideration of a GIS user's role. She comes up with a taxonomy of spatial integrity constraints with the three classes topological, semantic and user constraints which are each further classified as static or transitional. Semantic integrity constraints make use of the meaning of objects, keeping the user from inserting for example "a road running through any body of water" [5]. User defined integrity constraints go one step further and are used to represent specific domain knowledge. Static enforces that constraints are satisfied at any time ("height of a mountain may not be negative" [5]) whereas transition describes how the data may be changed ("height may not decrease" [5]). For the modeling of semantic constraints she proposes a knowledge base in order to be able to reason about the constraints and also to benefit from the object-oriented approach. That way a constraint can be modeled for a (abstract) supertype and inherited from all subtypes.

Improving spatial consistency of existing vector data sets is the aim of the work of Servigne et al. [21]. Three kinds of spatial consistency errors are identified, namely structural, geometric and topo-semantic errors. Structural errors are due to a mismatch between the data model and its storage, for example often tricks are used to store a polygon with holes. Because they depend on the data model, they cannot be specified in a general way. Geometric errors arise if shapes or positions, referred to as geometric attributes, of objects are invalid. Topo-semantic errors are related to the topological relations of semantically defined real world objects. The definition and implementation of the latter constraints follows the work of Ubeda and Egenhofer [23]. A general framework is presented that can be used for databases and an example is given for cadastral data. Servigne et al. also propose the reusage of geometric properties, as they commonly apply to data modeling. Checking processes should be evaluated regarding if all erroneous objects are detected and if legal objects are also detected. This concept can also be found in binary classification, where the classes are termed false negatives and false positives. Identified limitations of the framework of Servigne et al. are error tolerance and usage of exceptions. Because objects or object classes often differ in their accuracy, the error tolerance should be defined for each of them separately. Also it cannot be assumed that constraints always apply to all objects. Therefore a framework should permit exceptions for individual objects or subtypes.

Borges et al. [2] aim at a holistic approach to address the relationship between spatial information, relationships and integrity constraints and merge them into one data model called “OMT-G”. Concerning the classification of constraints they use the already listed results of Cockcroft [5]. A new aspect is the consideration of relationships that are closer to the mental model of the users, for example “near”, “north of”, and “in front of”.

Constraints are seen by Reeves et al. [19] from a different perspective. They use ideas from variational Computer Aided Design (CAD) models. There individual features are described by a so-called parametric model consisting of parameters and functions that derive primitives based on them. Reeves et al. list four main constructs for the parametric functions. Constraint-Based Definitions exploit explicitly defined spatial relationships between geometric items, for example perpendicularity. In contrast, Constraint-Based Virtual Definitions use higher order concepts, for example projection of one geometry onto another. Grapho-Numeric Expressions permit the usage of geometric items as arguments for functions, for example the distance between two individual items. The last construct are Numeric and Boolean Valued Expressions. The application domain for which the constraints are modeled are road traffic markings, e.g. pedestrian crossings and stop lines.

The specification and implementation of constraints for a geo-virtual reality system used for landscape modeling is shown by Louwsma et al. [12]. The classification of Cockcroft [5] is extended by a classification in another five subclasses. These are (1) number of involved objects/classes/instances (single vs. multiple instance of one vs. multiple classes), (2) type of properties and relationships involved (metric, topological, temporal, thematic, complex), (3) dimension (spatial, temporal, thematic, and mixed), (4) must never vs. must always, and (5) theorem-based (physically impossible) vs. design-based. Some research questions remained open according to Louwsma et al. Amongst others these are constraint checking, that is identifying if constraints conflict with one another, and the question of extendibility to 2.5D and 3D objects.

Besides the aforementioned virtual reality system three additional scenarios are presented by van Oosterom [15]. This includes the Dutch cadastral data, for which more than 50 constraints have been defined in several categories, including temporal constraints. For modeling topographic data the classification of constraint types differs yet again. There the five types of (1) single entity and single attribute, (2) single entity and multiple attributes, (3) geometry, (4) topology, and (5) relationship were identified. The third scenario covers the split of a parcel using a transactional Web Feature Service (WFS-T), whereby topology constraints as well as the question of atomic transactions have to be handled.

2.2 Constraints in generalization

A comprehensive constraint analysis for application in automated map generalization has been carried out by the Department of Geography of the University of Zürich as part of the AGENT project [8]. Constraints are used as a specification of which design specifications the solution (the result of the generalization process) should meet. In the report constraints are classified into five categories: (1) graphic constraints are concerned with the legibility of the map and therefore affect mainly minimal requirements that have to be satisfied for feature and symbol geometry, e.g. size, (2) topological constraints, (3) structural (or semantic) constraints which “describe higher order concepts and involve less explicit relationships” [8] and include diverse concepts such as alignment, size relationships, and spatial distribution but also the semantic of objects called “logical context”, e.g. rivers are not permitted on hilltops, (4) the more abstract Gestalt constraints relating to aesthetics and aspects of perception, and (5) procedural constraints related to the involved generalization process steps.

Besides the classification also several key aspects of constraints are identified in the report [8]. Constraints can specify situations to maintain or to avoid and often can be equivalently expressed either way. The satisfaction of constraints may be absolutely required (also referred to as “hard” con-

straints in the literature), being subject to optimization (termed “soft” or “weak”), or also being both, that is if absolute boundaries are defined but in these boundaries the optimal solution shall be found. If a constraint considers only one state of an object it is called intrinsic, if it compares the state with a prior one then it is called extrinsic. Another important aspect is the scope or extent of a constraint. Spatial scope deals with the neighborhoods where the constraint is applied, that is within a feature, between features, between feature classes, or within regions. In contrast, contextual scope deals with the priority of constraints and the circumstances under which they may be eventually relaxed. Constraints can consider only one object (called independent) or consider relations between objects (called contextual). Nevertheless, constraints often affect one another and a solution must consider their priority and their severity. Severity describes the deegree to which a constraint is violated. In order to establish the balance between priority, severity, and scope the usage of a constraint management system is proposed.

In the AGENT project [8] also modeling issues concerning constraints have been addressed. An important task is to discover the structures that are needed for constraint definition, for example the explicit modeling of neighborhood, analysis of alignments, or the classification of features according to parameters of their geometry or topology, e.g. sinuosity of rivers or roads.

Burghardt et al. [3] extended respectively changed the classification defined in the AGENT project. Application of their project is the formalization of constraints for common generalization tasks of several European National Mapping Agencies. The top level of their constraint topology differentiates between legibility constraints and preservation of appearance. However, more important is the different view on semantic aspects and the distinction by geometry type. Semantic aspects are not seen as an extra class. In contrast they can be combined with every other class and are therefore relevant for all of them. Geometry type distinguishes if a constraint affects point, line, or area features.

A comprehensive inventory of map objects relations for topographic maps is given by Steiniger and Weibel [22]. The typology includes only horizontal relations, whereas update and vertical relations are omitted. Horizontal refers to relations existing at the same scale, whereas vertical refers to relations between different scales. Scale here refers to map scale, resolution and level of detail in the same manner. Horizontal relations exist in both data integration and generalization tasks, whereas vertical relations are addressed by generalization. Update relations refer to changes of map objects in time. The typology is based on several research projects, including the work from the AGENT project.

3 MODELING AND ENFORCING CONSTRAINTS

The approaches for modeling and enforcing constraints are as diverse as their classification and usage. The different approaches are shortly described in this section and examples for, respectively references of, their usages are given.

Widely-used is the simple approach of *implicit constraint definition in the data model*, described in more or less detail in the corresponding documentation. There constraints are part of the definition of object classes and their enforcement is not specified. Prominent examples are the standards Geographic Information – Spatial Schema (ISO 19107), the Geography Markup Language (GML), X3D, and the Industry Foundation Classes (IFC). ISO 19107 provides “conceptual schemas for describing and manipulating the spatial characteristics of geographic features” [11] which includes definitions for geometry, topology and spatial operations. Constraints concerning geometry are for example used for defining interpolation, or for stating that polygons must be coplanar. Topological constraints must be satisfied when complex objects are built from several object parts. The parts then have to be disjoint and have common boundaries (touch). Being built on base of the ISO 19107 standard, GML [16] brings in additional constraints in form of additional object classes. These are shared geometries

("XLinks") and the "Rectangle" class, which is conceptually a subtype of a polygon with five points and four right angles. However, it is not derived by restriction, but modeled independently due to restrictions of the XML Schema language. IFC [10], an open format specified for interoperability in the building industry, also defines several object classes having constraints, for example "IfcRectangularPyramid". The same applies to the X3D standard [25] used for modeling 3D objects.

From the presented research approaches Borges et al. [2] also use an implicit constraint definition. In their object-oriented data model for geographic applications called "OMT-G" constraints are implicitly defined by the proposed classes and their relationships. The actual integrity checking is then delegated to the implementation in a database which is not addressed by the authors.

An *explicit constraint definition in the data model* is shown by Reeves et al. [19]. Instead of extending the GML feature schema, which is the recommend approach for building application schemas based on GML, Reeves et al. produce a novel geometry schema with own geometry types that explicitly support constraints. Spatial relationships are modeled between "parent" and "child" geometries, whereby parents are composites of child geometries, for example include all stripes of a pedestrian crossing. The children on their part include the parameters needed for the constraint, for example the offset distance. According to Reeves et al. [19] the data can be exported to standard GML if all expressions are evaluated.

Van Oosterom [15] presents another approach for explicit constraint definition based on the scenario of modeling topographic data in the Netherlands. The constraints are specified along with the data model definition in an application specific XML format.

Constraint checking with database triggers is used by Louwsma et al. [12], Duboisset et al. [9], and van Oosterom [15]. Database triggers are small pieces of procedural code that can be used to automatically check certain conditions when data is inserted, updated or deleted. The mentioned authors use triggers to avoid insertion of invalid data and make therefore use of the spatial extensions of the database management systems.

Cockroft [6] stores *constraints as metadata in a repository*, which are then part of the database. The repository contains besides constraints on attribute values and spatial relationships also information about quality and lineage of data as well as its rendering. In the presented approach constraints are defined in a graphical user interface and translated automatically into data definition language statements defining parts of the database model or as queries for a GIS system.

This shows that the approach for the definition respectively modeling of constraints may differ from their enforcement. Cockcroft [6] uses for the enforcement the approach of *constraints as queries or procedural code in a GIS*. This approach is also used by Servigne et al. [21] which added constraint definition and error correcting interfaces as tools to a GIS.

Another approach is the *constraint definition in an ontology* presented by Mäs et al. [13]. Constraints for spatial relations are defined by the authors using a slightly extended version of the Semantic Web Rule Language (SWRL). The ontology, built up for a landslide application, also contains correction instructions given in natural language. These are part of a decision support system guiding field users during data acquisition.

As part of the OGC Web Services, Phase 4 (OWS-4) Testbed topological constraints were also modeled in a language following the SWRL [18]. Therefore data has to be mapped first from GML into an internal schema in order to be checked.

A later publication for the landslide application and from the same research group proposes the use of *Constraint Decision Tables (CDT)*. Wang et al. [24] use CDTs to formalize constraints and correction instructions by means of a so-called Event-Condition-Action (ECA) rule. For an event like

an update the conditions to check, and if they apply, the corresponding actions to take are formulated. The proposed action can also be a list of alternatives. The CDTs are converted to XML documents in the RuleML language and enforced by an independent software component.

The last approach to be presented is the *constraint definition based on the Object Constraint Language (OCL)*. OCL [14] is part of the Unified Modeling Language (UML) which is the preferred language for modeling conceptual schemas. For example, UML is also used in the standards GML and WPS from the Open Geospatial Consortium. OCL is a formal language used to describe constraints about objects in UML models unambiguously. OCL has been proposed for the constraint specification of geo objects in several research projects, for example by Casanova et al. [4], Duboisset et al. [9], Louwsma et al. [12], and van Oosterom [15]. Important to note is also, that OCL is used in the AFIS-ALKIS-ATKIS Reference Model (AAA) as part of the German SDI [20]. The AAA model has been designed for the basic geodata sets of all public surveying and mapping authorities in Germany. Aiming at GIS interoperability, its application schema is based on ISO and OGC standards and modeled in UML.

In the following section we present why, from our perspective, OCL is the most promising of the presented approaches.

4 OCL AND GEOOCL

Starting with a short overview of OCL, the approaches of the mentioned research projects and the AAA model are shown in this section. Then we highlight the advantages of using OCL instead of, or as part of, other alternative approaches for modeling constraints as presented in Sect. 3. We also show the versatility of OCL concerning the enforcement of the defined constraints. Afterwards our proposal of a Geo Object Constraint Language (GeoOCL) is presented and analyzed regarding the requirements discussed in Sect. 2.

4.1 Short overview of OCL

OCL [14] can be used for several purposes, amongst others to specify invariants on classes and types in UML models. Invariants correspond exactly to constraints, because they "must be true for all instances […] at any time" [14]. The basic OCL syntax is shown by the following example, which specifies that all streets must have a maximum horizontal grade of 8 percent.

```
context Street
inv:  self.horizontalGrade <= 8
```

The term after the keyword `context` denotes the type, class, interface, association or data type to which the expression applies. The keyword `self` then refers to one instance of the context. The label `inv:` declares that an invariant constraint shall be used. In the term in the second line the value of the attribute `horizontalGrade` of a particular instance referenced as `self` is evaluated. The Boolean term must be true for all street instances.

OCL has built-in support for Boolean, Integer, Real, String and Collection types including common operations on them. Also new operations can be defined in an OCL document. The next example shows some more sophisticated aspects of OCL.

```
context Street
inv:  Street.allInstances()->exists(name = 'Main street')
inv:  self.country <> 'DE' implies self.maxSpeed <= 130
inv:  Street.allInstances()->isUnique(name)
```

The Street.allInstances returns the set of all streets that exist. In the first invariant the constraint is enforced that at least one object with the given name exists. The second invariant shows how constraints can be enforced only on some instances with <> meaning "not equal". So only if the term be-

fore `implies` is true, then the term after it is evaluated. Boolean expressions can also be formulated by `if-then-else-endif` constructs and several of them can be concatenated by `and` statements. The last invariant finally ensures that street names are unique.

4.2 Approaches using OCL

Louwsma et al. [12] actually only propose the use of OCL. As already mentioned before, their practical implementation is the direct specification of database triggers. However, they give detailed information on the possible OCL usage in their scenario. They also conclude, that "in future development environments it should be possible to generate the […] application code that implements the constraints (as specified in UML/OCL) automatically" [12]. The problem they see is that "too more research is needed into the best way of incorporating integrity constraints in XML schemas" [12], such as GML. We will address the XML Schema aspect in the next paragraphs. Nevertheless, also a direct solution is possible, which has been shown by Duboisset et al. [9] as explained later.

With an XML Schema [26] a set of rules for the validity of an XML document can be specified. With these rules for example valid datatypes e.g. double or string, legal range of values e.g. enumerations or maximum and minimum values, and number of items of a collection can be defined. The following example is equivalent for the already presented restriction in OCL if the datatype `horizontalGradeDouble` is specified for the attribute or element `horizontalGrade` in the XML Schema.

```
<xsd:simpleType name="horizontalGradeDouble">
  <xsd:restriction base="xsd:double">
    <xsd:maxInclusive value="8"/>
  </xsd:restriction>
</xsd:simpleType>
```

However, the constraints that are expressible with an XML Schema are rather limited. Additionally, most of them can be simply expressed by corresponding OCL expressions. Therefore we do not recommend transferring OCL constraints to XML Schema documents. By keeping them separate a more flexible solution is possible.

Duboisset et al. [9] use an extended version of the OCL2SQL tool in order to automatically generate database triggers that are equivalent to OCL for different relational database management systems. For being able to specify topological constraints, they extended OCL with the 9 Intersection Model (9IM). Therefore they integrated eight new operations, one for each topological relationship on regions. The operations can be used on geometries and evaluate to true if they adhere to the given topological relationship. Constraints can also be formulated for composite geometries and the number of parts implied in a topological relationship can be specified, for example that each part of one geometry meets two parts of another. Additionally, they extended their so-called OCL9IM with an adverb model permitting to add one of seven adverbs such as "entirely", "mostly" or "never" to the relationships. In order to show the approach of Duboisset et al. the following slightly shortened OCL statement is reproduced from their paper [9].

```
context DowntownBuildingLot
inv:  self.geometry->inside("mostlyRev", self.downtown.geometry)
```

As can be seen in the previous example the "downtown geometry" must be also part of the "building lot" class because the OCL expression is limited to its context. Here the work of Casanova et al. [4] offers a solution by another adoption of OCL. They used OCL for quality checking of datasets modeled in the Geographic Data Files (GDF) standard, which is used for road related information. Concerning the operations they used standard OCL, however they addressed other restrictions of OCL in their work. Their adaptations permit amongst others to express constraints on multiple classes, using parametric constraints and easier constraint composition. Parametric constraints refer to constraints that take one or more parametric types as arguments that can be used in the OCL expressions, which is comparable to C++ templates or JAVA generics.

The AFIS-ALKIS-ATKIS Reference Model [20] uses OCL mainly in ALKIS (Official Cadastral Information System). There alpha-numerical, geometric and topological constraints are expressed in OCL whenever possible. For example for land parcels several constraints have to be satisfied. Two land parcels are not allowed to intersect and the boundaries of a land parcel can either be specified by lines or circular arcs. Arcs have to be defined by three points which have to be given according to the orientation of the arc. The AAA Reference Model makes no proposal how the constraints can or should be implemented.

4.3 Advantages and versatility of OCL

In Sect. 3 several approaches for modeling constraints were presented and are analyzed in comparison to OCL in the following. Obviously, an implicit constraint definition in the data model is not feasible, because constraints have to be explicit in order to be checked.

OCL is part of the data model definition in a conceptual schema in UML. It seems therefore not practicable to include the constraints in every file that includes data being modeled based on the UML schema, as proposed by Reeves at al. [19]. If new constraints are added, updated or removed, all data files would have to be changed. The approach presented by van Oosterom [15] with the application specific XML file for both data model and constraints is not standard-based, which UML and OCL are.

As a specification language, OCL is independent from its actual implementation. Therefore, the level of abstraction is settled above actual implementations, such as database triggers ([9], [12], [15]), queries or procedural code in a GIS ([6], [21]), or constraints as metadata in a database repository ([6]).

The definition of constraints in an ontology ([13]) makes perfectly sense, especially concerning their re-usage. This way, constraints can be defined for concepts and not for distinct data models and their respective terminology. However, today's data models are rarely modeled as ontologies. But OCL is prepared for future developments, because it could be included, or automatically derived, from ontologies.

Constraint Decision Tables ([24]) define the actions that have to be carried out if constraints are violated. We see them as an extension to using OCL, because the actions depend on the application use case scenario. For the presented system for field data acquisition ([24]), guidelines for users have to be shown with correction instructions. However, if the data has to be inserted in a database other actions are needed. This means that the constraints should be defined in a fix manner, whereas based on the application scenario the actions differ. In OCL each invariant can be labeled with a name, which then could be reused in different CDTs.

OCL is also universal concerning the time when constraints are enforced. They can be checked e.g. upon insertion, update, or deletion of features as well as for complete existing datasets.

The versatility of OCL concerning the enforcement of constraints is shown in Fig. 1 along with the approaches by Reeves et al. [19] with GML as well as van Oosterom [15] with XML. The OCL definitions are part of the UML model, although they are often stored in a separate file. Implementations enforcing these constraints then have several options depending on the target environment. We use e.g. the original (Geo)OCL in order to check geodata directly within a JAVA application. However, OCL constraints are often translated, e.g. into database triggers defined in SQL as shown by Duboisset et al. [9] with their OCL2SQL tool, or into queries or procedural code in a GIS system.

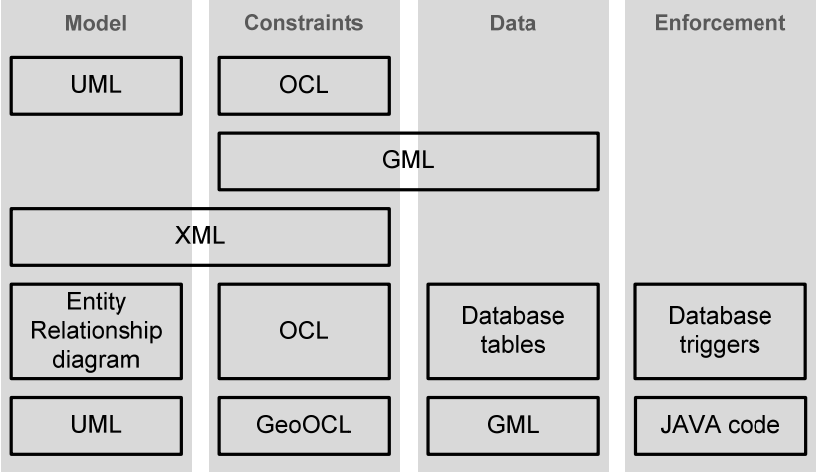

Figure 1: Different approaches for modeling and enforcing constraints.

4.4 GeoOCL

We have now provided the necessary background on OCL for presenting our research on creating a Geo Object Constraint Language (GeoOCL). In the following we provide evidence that OCL and our extension GeoOCL allow for flexible and expressive constraint formalization. Additional examples point out how to express some constraints.

In general terms, GeoOCL extends OCL mainly by the new data type Geometry and new operations on that data type. GeoOCL therefore spatially enables OCL. As an extension, all constraints defined in GeoOCL also apply to all sub-classes of the context they are defined in. The thematic scope of a constraint is already considered by the `implies` statement in OCL. GeoOCL follows the same approach by supporting spatial filter operations, as can be seen in the following example. The example also uses a local variable defined by the OCL construct `let-in` and a `if-then-else-endif` construct. In order to keep the example clean, the definition of a polygon as Well Known Text (WKT) in the parentheses is omitted here.

```
context Street
inv:  let specialDistrict : Geometry = Geometry.fromWKT() in
      if self.geometry->intersects(specialDistrict) then self.maxSpeed <= 30 else
      self.maxSpeed <= 130 endif
```

In more technical terms, the OCL grammar is extended to handle the new data type Geometry. Important to note is that Geometry is not a basic data type or literal such as String or Integer. Instead the data type follows the concept of a collection. Thereby the parts of a distinct Geometry can be accessed; e. g. for a LineString all lines with their respective start and end points can be referred to and therefore used in constraint expressions. This results also in the rule that all operations on Geometry are written with an arrow and not a dot notation. The operations defined for Geometry shall include all needed operations in the context of GIS. To date however only some are included, but based on previous research, constraint operations of different domains are collected. What is already considered, are operations for geometric, topologic and statistic constraints. Due to size limitations of this paper they cannot all be presented. However some examples for operations follow: geometric (area, length, perimeter, convexHull, collinear, distanceTo, direction), topologic (disjoint, touches, overlaps), and statistic (count, mean).

Constraints can be defined not only for each dataset in a separate OCL document, but also depending on the requirements of a specific application. This already addresses the first recommendation by Cockcroft [5], namely the consideration of user roles. Different OCL documents offer a flexible constraint checking for different users or application scenarios without changing the underlying data model. Also the three identified constraint classes topological, semantic and user defined can be mapped with OCL. Topological constraints can be handled with the approach of Duboisset et al. [9].

Following the classification of Servigne et al. [21] constraints for the two classes of geometric and topo-semantic errors can be specified in a general way. The validity of the geometry referring to shape and positions can be checked with the already presented methods or by the derivation of new OCL operations. Topo-semantic constraints can also be addressed by OCL. We have also shown that OCL supports exceptions for individual objects or subtypes, e.g. by using the `implies` statement.

An example for the need of defining new OCL operations is given by the constraints on breaklines, e.g. having minimum length or maximum curvature. The two operations `length` and `maxCurvature` (returning the maximum curvature in degrees) then have to be implemented by tools that check these constraints.

```
context Breakline
inv:  self.geometry->length() >= 10
inv:  self.geometry->maxCurvature() < 70
```

Borges et al. [2] proposed the use of relationships being closer to the mental model. We see the task of translating a concept from the mental model to the corresponding discrete range of values one step before the formal OCL definition. Saying so, a concept such as “north” can be translated e.g. to `315 < azimuth <= 45`.

The requirements concerning constraints mentioned by Reeves et al. [19] also can be mapped with OCL. Of course new operations have to be defined, e.g. for checking perpendicularity or higher order concepts such as projection. Grapho-Numeric expressions are also possible by using the respective attributes or objects as parameters for the new operations.

The classification of Louwsma et al. [12] requests the consideration of the number of involved objects/classes/instances. This can get quite complex, however with the extensions to multiple classes by Casanova et al. [4] and the consideration of the number of parts implied in a topological relationship presented by Duboisset et al. [9], we are confident that OCL can cope with that requirement too. Temporal logic is already considered by several OCL extensions. The other mentioned subclasses are also addressable with OCL, including extension to 2.5 and 3D. Identifying if constraints conflict with one another is out of the scope of a specification language such as OCL.

The following example for the constraint stating that noise abatement walls are disjoint from street geometries, points out how constraints on multiple classes can be expressed in GeoOCL following the notation of Casanova et al. [4].

```
context NoiseAbatementWall
inv:  Streets.allInstances()->forAll(s:Street|s.geometry->disjoint(self.geometry))
```

Concerning the categories specified in the AGENT project [8] only procedural constraints are not expressible, however Gestalt constraints could get quite complex. Procedural constraints are also not in the scope of our work, because we see constraints as a formulization of the solution and not of the way how it is reached. This makes processes exchangeable exactly as needed in the idea of Web Processing Services.

For now we didn’t address the requirements that are not met by a solution based on the OCL yet and are subject to further research. First of all, the dynamic aspects, referred to as transitional by Cockcroft [5] and also mentioned by most other authors, have to be taken into account. In OCL pre-

and postconditions of operations can be defined. If for all classes a (virtual) operation for the transition processes, such as data integration, is defined, then their pre and post-conditions can be specified. The (Geo)OCL expression for the already presented constraint requiring that the height of buildings may not be changed, but their size may differ by ten percent and they may be displaced in a limited range is shown in the following example. The variables followed by the keyword `@pre` refer to the state before the operation was executed, therefore making a comparison possible. This approach also allows for the specification of individual error tolerances for objects and object classes as requested by Servigne et al. [21].

```
context Building::integrate()
post: self.height = self.height@pre and self.size <= self.size@pre * 1.1 and
      self.size >= self.size@pre * 0.9 and self.geometry->
      distance(self.geometry@pre) <= 0.2
```

For the consideration of the two important aspects of severity and priority the existent OCL features are not sufficient. These two parameters should be defined for each constraint statement and could then lead to different values for the post-conditions. To date we cannot offer a solution for this problem, however it is on our research agenda. The scope of a constraint already can be specified with the presented OCL statements and adaptations such as the `implies` construct. A constraint management system then can be built that gets its information about severity, priority and scope of constraints solely from the OCL definition.

Also the different levels of satisfaction for constraints as mentioned by the AGENT project [8] have to be considered in further research. A constraint definition equivalent to the `inv` keyword for constraints that are absolutely required, could be defined for constraints that are subject to optimization, e.g. by introducing the keyword `opt`.

5 CONCLUSIONS

The formalization of spatial constraints has to answer several research questions. Firstly, the solution has to be able to cover requirements from different application scenarios and disciplines. In this context, it has been shown that requirements from the data integration and generalization community represent two sides of the same coin. Secondly, the formalization has to be standard-based in order to gain success. The proposed solution called GeoOCL spatially enables the Object Constraint Language (OCL). It has been shown to which extent the GeoOCL already satisfies the requirements and also open research questions have been addressed.

ACKNOWLEDGEMENTS

The research described in this paper is part of the GDI-Grid project (Spatial Data Infrastructure Grid) funded by the German Federal Ministry of Education and Research (BMBF). The support is gratefully acknowledged.

BIBLIOGRAPHY

1. Arbeitsgemeinschaft der Vermessungsverwaltungen der Länder (AdV) (2008) Dokumentation zur Modellierung der Geoinformationen des amtlichen Vermessungswesens (GeoInfoDok) (in German). AdV, version 6.0, see also http://www.adv-online.de, last date accessed 12/2008

2. Borges KAV, Davis CA Jr., Laender AHF (2002) Integrity Constraints in Spatial Databases. In: Doorn JH, Rivero LC (eds) Database integrity: challenges and solutions. IGI Publishing, Hershey, pp 144-171

3. Burghardt D, Schmid S, Stoter J (2007) Investigations on cartographic constraint formalisation. In: ICC 2007: Proceedings of the workshop of the ICA commission on generalisation and multiple representation, August 2-3, 2007, Moscow, Russia

4. Casanova M, Wallet T, D'Hondt M (2000) Ensuring quality of geographic data with UML and OCL. In: Goos G, Hartmanis J, Leeuwen van J (eds) <<UML>> 2000. Springer, Berlin Heidelberg, pp 225-239

5. Cockcroft, S (1997) A Taxonomy of Spatial Data Integrity Constraints. GeoInformatica 1 (4): 327-343

6. Cockcroft S (2004) The design and implementation of a repository for the management of spatial data integrity constraints. GeoInformatica 8 (1): 49-69.

7. Czerwinski A, Sandmann S, Stöcker-Meier E, Plümer L (2007) Sustainable SDI for EU noise mapping in NRW – best practice for INSPIRE. International Journal for Spatial Data Infrastructure Research (IJSDIR) 2007 (1): 90-111

8. Department of Geography, University of Zürich (1998) Constraint analysis. Technical Report. University of Zürich

9. Duboisset M, Pinet F, Kang M-A, Schneider M (2005) Precise modeling and verification of topological integrity constraints in spatial databases: from an expressive power study to code generation principles. In: Delcambre L, Kop C, Mayr HC, Mylopoulos J, Pastor O (eds) Conceptual Modeling – ER 2005. Springer, Berlin Heidelberg, pp 465-482

10. International Alliance for Interoperability (IAI) (2007) Industry Foundation Classes. IAI Specification, IFC2x Edition 3

11. ISO (2003) Geographic Information – Spatial Schema. ISO Standard, ISO 19107:2003

12. Louwsma J, Zlatanova S, Lammeren R van, Oosterom P van (2006) Specifying and implementing constraints in GIS – with examples from a Geo-Virtual Reality System. GeoInformatica 10 (4): 531-550.

13. Mäs S, Wang F, Reinhardt W (2005) Using ontologies for integrity constraint definition. In: Proceedings of the 4th International Symposium on Spatial Data Quality'05, August 25-26, Beijing, China

14. Object Management Group (2006) Object Constraint Language. OMG Available Specification, version 2.0

15. Oosterom P van (2006) Constraints in spatial data models, in a dynamic context. In: Drummond J, Billen R, Joao E, Forrest D (eds) Dynamic and mobile GIS: investigating changes in space and time. Taylor & Francis, London, pp 104-137

16. Open Geospatial Consortium (2007). OpenGIS Geography Markup Language (GML) Encoding Standard. OpenGIS Standard, OGC 07-036, Version 3.2.1, see also http://www.opengeospatial.org/standards/gml, last date accessed 12/2008

17. Open Geospatial Consortium (2007) OpenGIS Web Processing Service. OpenGIS Standard, OGC 05-007r7, Version 1.0.0, see also http://www.opengeospatial.org/standards/wps, last date accessed 12/2008.

18. Open Geospatial Consortium (2007) OWS4 – Topology quality assessment interoperability program report. Open Geospatial Consortium discussion paper, OGC 07-007r1, see also http://www.opengeospatial.org/standards/dp, last date accessed 12/2008

19. Reeves T, Cornford D, Konecny M, Ellis J (2006) Modeling geometric rules in object based models: an XML / GML approach. In: Riedl A, Kainz W, Elmes GA (eds) Progress in Spatial Data Handling. Springer, Berlin Heidelberg, pp 133-148

20. Seifert M (2006) AAA – the contribution of the AdV in an increasing European Spatial Data Infrastructure – the German way. In: Proceedings of the XXIII FIG Congress "Shaping the Change", October 8-13, Munich, Germany

21. Servigne S, Ubeda T, Puricelli A, Laurini R (2000) A methodology for spatial consistency improvement of geographic databases. GeoInformatica 4 (1): 7-34

22. Steiniger S, Weibel R (2007) Relations among map objects in cartographic generalization. Cartography and Geographic Information Science (CaGIS) 34 (3): 175-197.

23. Ubeda T, Egenhofer MJ (1997) Topological error correcting in GIS. In: Scholl M, Voisard A (eds) Proceedings of the 5th International Symposium on Advances in Spatial Databases. Springer, London, pp 283-297

24. Wang F, Reinhardt W (2007) Extending geographic data modeling by adopting constraint decision table to specify spatial integrity constraints. In: Fabrikant SI, Wachowicz M (eds) The European information society: leading the way with geo-information. Springer, Berlin Heidelberg, pp 435-454

25. Web3D Consortium (2004) Information technology – Computer graphics and image processing – Extensible 3D (X3D). ISO/IEC Standard, ISO 19775:2004

26. World Wide Web Consortium (2004) XML Schema Part 2: Datatypes Second Edition. W3C Recommendation