

Attribute grammar for 3D city models

J. Schmittwilken, D. Dörschlag, L. Plümer

Institute of Geodesy and Geoinformation, University of Bonn, Germany, {schmittwilken, doerschlag, pluemmer}@igg.uni-bonn.de

ABSTRACT: In this paper we show how attribute grammar formalism can be used as a modeling language for three dimensional geometric models. We focus on 3D building models and their decomposition into building parts. The specification of aggregation and the inheritance of shared parameters is discussed. The focus is on geometrical, topological and semantical constraints which govern aggregation. We show that whereas aggregation is a natural concept within context-free grammars, inheritance of parameters and constraints can neatly be expressed by semantic rules of attribute grammars. Semantic rules are used to specify form parameters and to inherit location parameters, to encode spatial operations, to represent symmetry and to specify topological constraints. We give an XML schema for spatial attribute grammars, an editing and processing tool and illustrating example results.

1 INTRODUCTION

Recent methods for the reconstruction of detailed 3D building models base on either LIDAR or image data of terrestrial or air-born origin. Automatic reconstruction, however, has not been achieved up to now. Overall there are two common approaches to interpret the observations: the top-down and the bottom-up method. In principle the latter detects atomic features like straight lines or planes and aggregates them to complex objects like roofs or stairways. Top-down methods use semantic models or ontologies to describe the whole object, often by aggregations, and then try to fit its parts into the observations.

In general, formal grammars are a natural candidate for the integration of bottom-up and top-down approaches. They have been introduced in order to generate well-formed sentences (objects) and to reconstruct (parse) the internal structure of well-formed objects by the same formalism. Han & Zhu (2005) discuss the role of formal grammars in computer vision and present a mixed top-down bottom-up approach with attribute graph grammars.

In this paper we show how attribute grammars can be used as a semantic model for 3D building models. Man made constructions like buildings and building parts are almost characterised by geometric invariants such as parallelism, perpendicularity and symmetry, especially mirror symmetry. Therefore we focus on the specification of geometric attributes and their derivation in a way which ensures that geometrical invariants, topological constraints and symmetry is preserved. Our notion of topology is based on the tessellation property which essentially means that geometric objects are both covering and intersection free. In other words, from the well-known ‘Egenhofer relations’ of topology (Egenhofer & Franzosa 1991) only *meets* and *disjoint* are allowed between different objects. Furthermore we present a representation of mirror symmetry which utilises the fact that while the location parameter of related geometric objects differ their form parameters are identical (up to symmetry).

The importance of parameter inheritance between 3D objects and their parts can best be illustrated by a simple example. Stairs are made of steps. Within a stair the height and depth of steps are invariant. Entrance Stairs are almost always either perpendicular or parallel to the facade of

the building. The facade is almost parallel to the centre line of the street. Inheritance of geometrical attributes may be direct and simple as in this case more complex in other examples, but, as a rule, it characterizes the relation between an object and its context. Whereas context-free grammars, which most often are referred to in computer vision, specify aggregation directly by so called production rules, attributed grammars augment production rules by semantic rules which specify constraints on attributes.

The following sections are structured as follows. After reviewing related work in the field of reconstruction of 3D building models and attribute grammars in Section 2, we discuss the major aspects of the incorporation of geometry into attribute grammars in Section 3. In section 4 our XML based framework for encoding grammars and a prototype for the generation and visualisation of 3D models is presented. Finally we discuss our results, draw a conclusion, and point out future work in Sections 5.

2 RELATED WORK

Formal grammars have been introduced by Chomsky (1956, 1959) to describe the linguistic structure of the natural and formal languages. He defined grammars as repeated transformations of phrase structures. The classification of formal grammars in four levels (type0 – type3) has been founded on his work and is still known as Chomsky hierarchy. In the sense of Chomsky a formal Grammar is defined as a quadruple $G = \{T, N, S, P\}$ of a finite set of terminal symbols T , a finite set of non-terminal symbols N with $T \cap N = \emptyset$, a start symbol S with $S \in N$ and a finite set of production rules P .

About a decade after Chomsky's fundamental work Knuth (1968, 1971) described how to assign semantics to context-free languages (Chomsky type2) by attribute grammars. His work was motivated by the formal design of compilers for typed computer languages, where the declaration of a variable at the beginning of a program has to be semantically related to an occurrence of that variable in an assignment a couple of code pages later. In order to express 'semantical' relations of that kind, a finite set of attributes $A = \{a_1, a_2, \dots, a_n\}$ is added to each symbol Y with $Y \in (N \cup T)$. The processing of the attributes in the derivation tree is defined by a finite set of semantic rules $R(P)$ attached to each production rule P .

Attribute grammars have been used in a variety of application (Alblas & Melichar 1991), particularly in programming language specification (Paakki 1995), and compiler design (Aho et al. 1999). In compiler design attributes can either be inherited or synthesised. Typically they are used for ensuring type conformity among separated symbols i.e. the declaration and the instantiation of a symbol. We use attributes for the information flow between separated nodes of the derivation tree. In contrast to the usage of attribute grammars in compiler design we do not restrict semantic rules on symbols but heavily use numbers and functions for the coding of geometrical relations or probabilistical information.

We shall also refer to clause grammars (Russell & Norvic 2003) which explicitly apply unification as a bi-directional communication channel which supports both top-down and bottom-up propagation of parameter values. The traditional distinction between inherited or synthesised attributes is not necessary in this context.

Duarte (2002) presents the discursive grammar i.e. a combination of shape grammar, description grammar and a set of heuristics. The presented grammar operates on shapes and uses split rules to perform predefined tessellations of the base shapes. The main disadvantage of this type of grammar is the inability of extending or rotating shapes.

The split grammar defined by Wonka et al. (2003) operates on shapes as well. Split grammars base on the concepts of Duarte and are used to generate building designs in the form of facades. Attributes associated to the shapes are used to store information about appearance on one hand and to control the derivation on the other hand. Wonka et al. use split grammars for a 2D description of facades.

Müller et al. (2006) extend split grammars for procedural modelling of buildings. They present a 3D grammar which implements affine transformations such as rotation. Symbols are represented by volumetric objects which interact like in constructive solid geometry (CSG, Män-

tylä 1988). They use snap lines to specify regularities between related objects on the visual surface of the facade like windows.

Larive & Gaildrat (2006) also present a split grammar based approach. They focus on the generation of facades as an aggregation of so called wall panels.

None of the cited concepts ensures the topological correctness of building models. All of them deal with intersection of shapes like CSG where intersection faces or lines of two volumetric shapes are not represented explicitly.

Ripperda & Brenner (2006) use formal grammars in combination with reversible jump Markov Chain Monte Carlo (rjMCMC) methods for the reconstruction of facades from terrestrial LIDAR or image data. They generate model assumptions which optimally fit the observations by guiding the derivation process by rjMCMC. However their work is limited to 2D representation of facades.

Müller et al. (2007) also use terrestrial images for a grammar based reconstruction of facades. They present a four step top-down method which ends up in a 3D facade model. The main failure cases described in their work are caused by irregularities on the facade.

Dörschlag et al. (2008) present a grammar based approach for the stepwise generation of model assumptions in a reconstruction process. They focus on attribute grammars derived from an ontology. Schmittwilken et al. (2007) also discussed the derivation of attribute grammars from semantic models i.e. models formulated in UML and OCL, the Unified Modeling Language and its Object Constraint Language.

3 ATTRIBUTE GRAMMARS FOR 3D OBJECTS

This section explores the potential of attribute grammars for geometric objects and shows how it can be used as a modelling language for 3D building models.

A special characteristic of the human ability of recognition is the concept of compositionality (Geman et al. 2002). Humans typically represent objects as aggregations of their parts. This is fundamental to language. How else should we know whether ‘the precipitous bank at the river’ is for flood control or for taking money into. Furthermore compositionality is an often used concept in 3D city modelling. CityGML which has been developed by the SIG 3D of North Rhine Westphalia and has strongly motivated this paper (OGC 2008, Gröger et al. 2005) is now the Open Geospatial Consortium’s international standard for 3D urban models. CityGML uses aggregation in addition to generalisation as a main modelling concept e.g. the CityGML schema defines that ‘a *CityModel* consist of *_CityObjects*’ and ‘a *Building* consists of *BuildingParts*’.

In addition to aggregation ‘inheritance’ is another important concept in urban modelling. Here we speak about the ‘inheritance of parameters’ in contrast to the ‘inheritance of attributes and methods’ in object-oriented programming. Parameters are propagated between the whole object and its parts. For example the relation of the orientation of two objects can be inherited. The major types of orientation relations are ‘is parallel’ or ‘is perpendicular’. This will be illustrated by some examples in the following of this paragraph.

It is not difficult to observe that – as a rule – facades are parallel to the centreline of the street. As a rule, buildings inherit their orientation from the street. Other Building parts inherit their direction from the general orientation of the aggregated building, particularly from the front facade. Almost always, the entrance stair is perpendicular or parallel to the front facade, almost every apex is parallel to its facade, and every dormer apex is perpendicular to the apex of the whole roof.

The concepts of aggregation can nifty be expressed by context-free grammars. Each production rule represents an aggregation: the left hand side symbol of the rule forms the whole object which is aggregated by its parts i.e. the right hand side symbols of the production. E.g. the German norm for staircases (DIN 18065) defines a flight as an unbroken series of steps which consist of riser and tread respectively. Figure 1a shows an UML diagram for this model of a flight. Table 1 shows the production rules of the according grammar. Capitalised words denote non-terminal symbols and lower case words terminal symbols. The grammar of Table 1 matches the semantics directly, but in view of the following paragraph it is convenient to simplify it. In production rules *P1* and *P2* the non-terminal *Step* can be replaced by the terminal symbols *riser* and *tread* and the production rule *P3* can be cancelled. We denote the whole as parent and its

parts as children. This metaphor, which refers to the derivation tree (Fig. 1b), is common in the context of formal grammars.

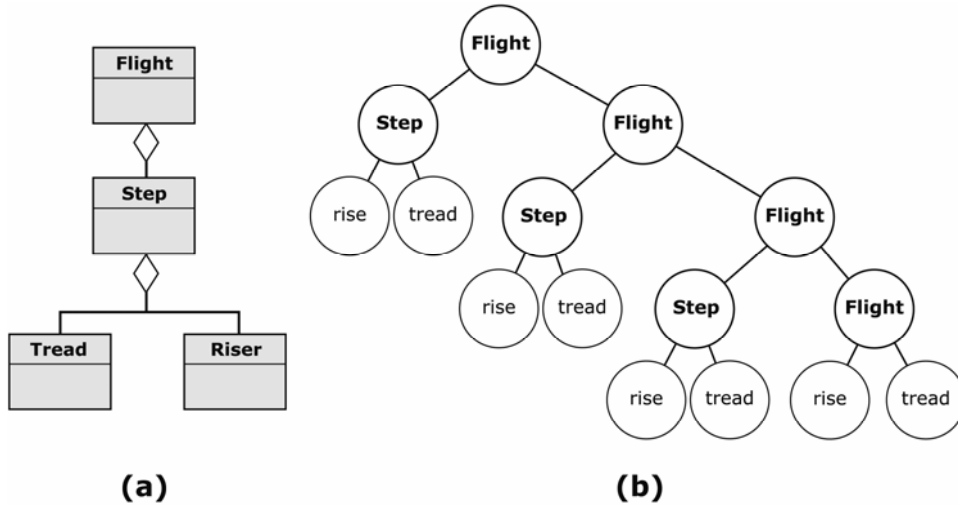


Figure 1. UML diagram of a flight (a) and a derivation tree (b).

Table 1. A set of production rules for flights

$P1: Flight \rightarrow Step\ Flight$

$P2: Flight \rightarrow Step$

$P3: Step \rightarrow riser\ tread$

3.1 Geometry

The concept of inheritance can neatly be implemented by attribute grammars. As mentioned above parameters are propagated from the whole object to its parts or in terms of the grammar from left hand side symbols of a production rule to right hand side symbols i.e. from parents to their children in the derivation tree.

In the following we will focus on location parameters and form parameters which give the geometric description of the model. Furthermore the geometry specifies and constrains the aggregation of objects. The propagation of location and form parameters is essential for the consistency of 3D building models.

Resuming the previous example we require all steps within a stair to have the same rise, tread depth, and width. We also require that neighbouring steps are connected and that no two different steps overlap. The identity of form parameters and the connection of risers and treads specifies the aggregation and can be specified by semantic rules. Figure 2a shows the extended UML diagram with the *Step* class skipped. The semantic rules corresponding to the modified production rule $P1$ are shown in Table 2. Rules $R1$ to $R4$ propagate form parameter from the flight to rise and tread.

To propagate the location parameter we introduce the reference point concept. The attribute *refPoint* is attached to any symbol. It defines the orientation and the position of the local coordinate system relative to the global coordinate system of the whole model. Hence *refPoint* is an 6-tuple $\{x, y, z, \omega, \phi, \kappa\}$. Here x, y and z are three-dimensional coordinates, ω, ϕ, κ specify rotations around the x -, y - and z -axis. If required parameters for scaling and shearing could be added.

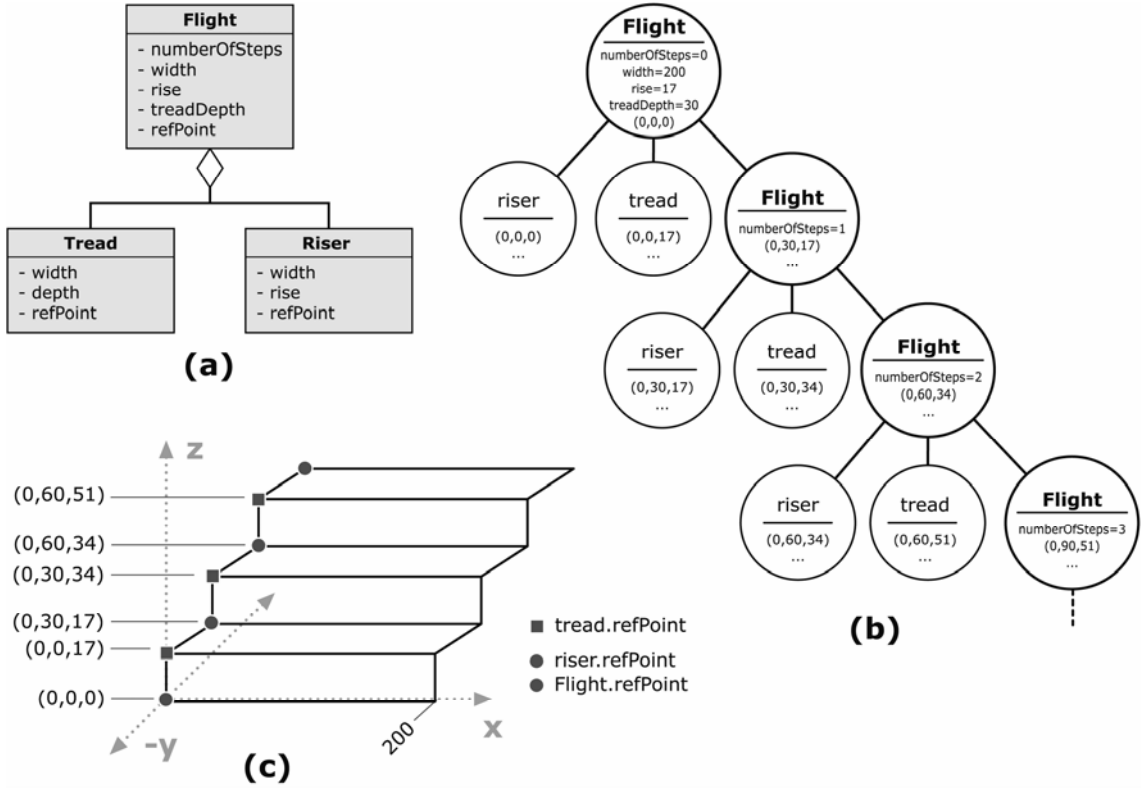


Figure 2. UML diagram with attributes (a), an exemplary derivation tree (b), and a sketch of geometrical interpretation (c).

Table 2. Semantic rules for production rule P1. The right hand side Symbol *Flight* is denoted by *Flight^l* to be distinguishable from *Flight*. All identity rules like *Flight^l.width = Flight.width* are omitted.

| | |
|----------------|--|
| <i>P1:</i> | <i>Flight</i> → <i>riser tread Flight^l</i> |
| <i>R1(P1):</i> | <i>riser.width := Flight.width</i> |
| <i>R2(P1):</i> | <i>riser.rise := Flight.rise</i> |
| <i>R3(P1):</i> | <i>tread.widht := Flight.width</i> |
| <i>R4(P1):</i> | <i>tread.depth := Flight.treadDepth</i> |
| <i>R5(P1):</i> | <i>tread.refPoint := transform(Flight.refPoint, 0, 0, Flight.rise, 0, 0, 0)</i> |
| <i>R6(P1):</i> | <i>Flight^l.refPoint := transform(Flight.refPoint, 0, Flight.treadDepth, Flight.rise, 0, 0, 0)</i> |
| <i>R7(P1):</i> | <i>Flight^l.numberOfSteps := Flight.numberOfSteps + 1</i> |

R5 and *R6* (Tab. 2) show that the inheritance of the *refPoint* attribute bases on the *transform()* function which implements the multiplication of the reference point with a homogeneous transformation matrix. The matrix and the corresponding call of the function is shown for a rotation around the *z*-axis in equation 1.

$$\begin{bmatrix} \cos \phi_z & -\sin \phi_z & 0 & \Delta x \\ \sin \phi_z & \cos \phi_z & 0 & \Delta y \\ 0 & 0 & 1 & \Delta z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} \quad (1)$$

$$\text{transform}(r, \Delta x, \Delta y, \Delta z, 0, 0, \phi_z) = r'$$

Storing the geometry in the attributes specifying the inheritance and further constraints by semantic rules has one important impact on the language defined by the grammar: It makes all basic assumptions on the geometric structure of the given objects explicit. It is not the sequence of symbols which governs the aggregation. In fact the left to right order in production rules and derived tokens is not relevant.

3.2 Symmetry

In architecture symmetry - especially mirror symmetry - is an ubiquitous design pattern. Facades of multi-storey buildings are highly symmetrical. As a rule windows are horizontally grouped and aligned in single rows and vertically in column. Facades of smaller buildings like residential houses are symmetrical as well (cf. Fig 3a). Building parts again follow the rules of symmetry e.g. double flight staircases (Fig. 3b) or the alignment of columns and arches in arcades (colonnades). The symmetry feature has been used for reconstruction of facades by several authors (Müller et al. 2007, Ripperda & Brenner 2006).

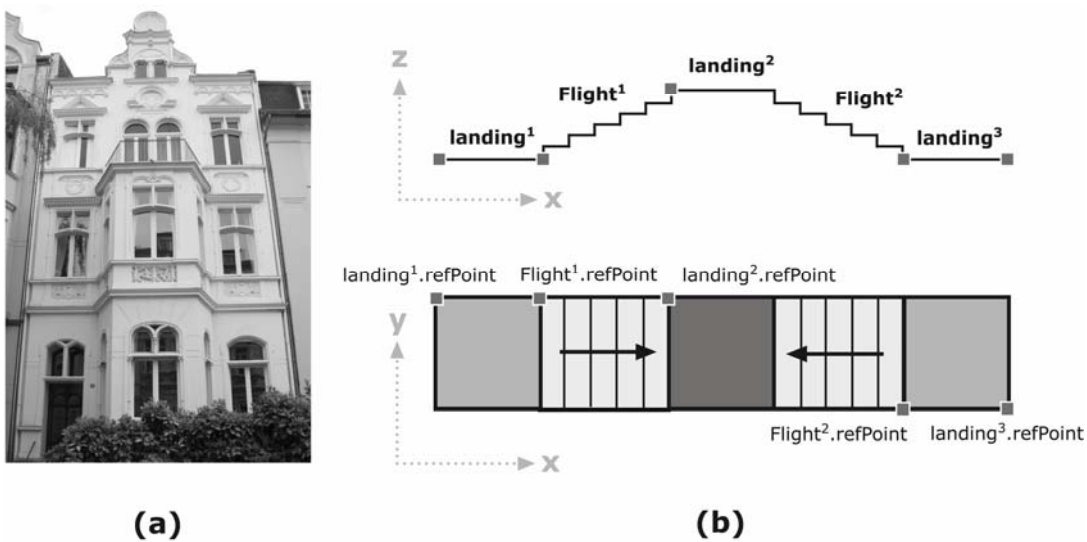


Figure 3. Image of a symmetrical facade (a), ground plot of a mirror symmetrical double flight staircase.

Table 3. Production rule and an extract of semantic rules for a double flight staircase. The omitted semantic rules for landing2, Flight2 and landing3 are analogously.

| | |
|-----------|---|
| $P4:$ | $StairCase \rightarrow landing^1 Flight^1 landing^2 Flight^2 landing^3$ |
| $R1(P4):$ | $landing^1.depth = StairCase.depth$ |
| $R2(P4):$ | $landing^1.width = StairCase.width$ |
| $R3(P4):$ | $Flight^1.width = StairCase.width$ |
| $R4(P4):$ | $Flight^1.rise = StairCase.rise$ |
| $R5(P4):$ | $Flight^1.treadDepth = StairCase.treadDepth$ |

Symmetrical objects differ in their location parameter whereas their form parameter are similar (identical up to symmetry). In attribute grammars this is implemented by semantic rules. The double flight staircase shown in Figure 3b is generated by $P4$ (Tab. 3). The symmetry is triggered by the inheritance of form parameter (semantic rules in Table 3) on one hand and by the transformation of location parameter on the other hand. The mirroring of geometrical representation is shown in Table 4. The mirrored symbols are rotated by π . Therefore the reference point has to be translated in direction of negative y-axis by the width of the tread. Note that different occurrences of the same non-terminal symbol are differentiated by superior indices.

Table 4. Semantic rules for the transformation of the reference points.

| | |
|------------|---|
| $R6(P4):$ | $landing^1.refPoint = transform(StairCase.refPoint, 0,0,0, 0,0,0)$ |
| $R7(P4):$ | $Flight^1.refPoint = transform(StairCase.refPoint, landing^1.depth, 0, 0, 0, 0, 0)$ |
| $R8(P4):$ | $landing^2.refPoint = transform(Flight1.refPoint,$ $Flight1.numberOfSteps * Flight1.treadDepth, 0,$ $Flight1.numberOfSteps * Flight1.rise, 0, 0, 0)$ |
| $R9(P4):$ | $Flight^2.refPoint = transform(Flight^1.refPoint,$ $landing^2.depth + 2 * Flight^1.numberOfSteps * Flight^1.treadDepth, -Flight^1.width, 0, 0, 0, \pi)$ |
| $R10(P4):$ | $landing^3.refPoint = transform(landing^1.refPoint,$ $landing^2.depth + 2 * landing^1.depth + 2 * Flight^1.numberOfSteps * Flight^1.treadDepth, -$ $landing^1.width, 0, 0, 0, \pi)$ |
| $R11(P4):$ | $Flight^1.numberOfSteps = Flight^2.numberOfSteps$ |

The previous grammar is a special case where the symmetry is generated by a single production rule. The transformation of the reference point in a more general way is shown in Table 5. There we use a palindromic context-free grammar with a recursive rule. See Figure 4 for a geometrical illustration.

The following assumptions are made (cf. Fig. 4): (a) The axis of symmetry is parallel to the y-axis, (b) the generated objects consist of axially parallel rectangles or cuboids, and (c) adjacent objects have a shared face parallel to the y-z-plane. The illustrations however restrict themselves to a projection on the x-y-plane. All other cases are analogously. Mention that the semantic rules operate on non-terminal symbols with unknown geometrical dimension. For that reason the general attributes $dimX$, $dimY$ and $dimZ$ are introduced for the maximum dimension of any object. Their values are derived when the derivation tree branch of the non-terminal symbol terminates. The attributes $offsetY$ and $offsetZ$ give the translation in y and z direction. The universal attribute $formParameter$ of $R4$ and $R5$ subsumes all form parameter attributes.

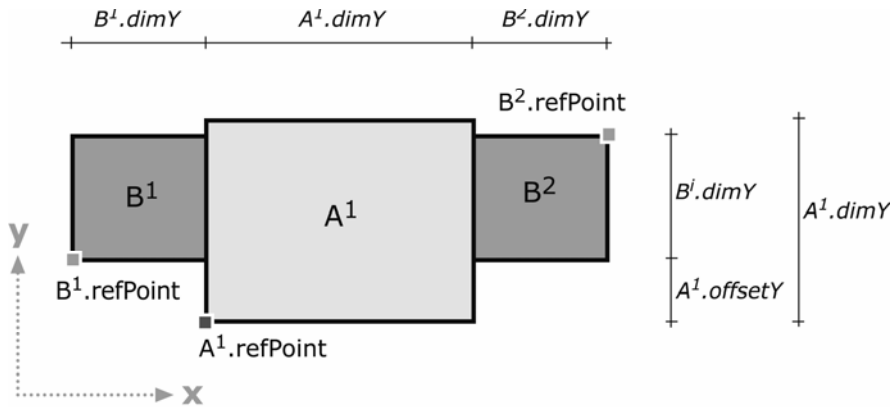


Figure 4. Geometrical representation of a palindromic context-free grammar rule.

Table 5. General implementation of mirror symmetrical transformation of reference points by a palindromic recursive production rule. The indices are used to differentiate symbols.

| | |
|-----------|--|
| $P5:$ | $A \rightarrow B^1 A^1 B^2$ |
| $R1(P5):$ | $B^1.refPoint = transform(A.refPoint, 0,0,0, 0,0,0)$ |
| $R2(P5):$ | $A^1.refPoint = transform(B^1.refPoint, B^1.dimX, A^1.offsetY, A^1.offsetZ, 0, 0, 0)$ |
| $R3(P5):$ | $B^2.refPoint = transform(B^1.refPoint, 2*B^1.dimX+A^1.dimX, -B^1.dimY, 0, 0, 0, \pi)$ |
| $R4(P5):$ | $B^1.formParameter = B^2.formParameter$ |
| $R5(P5):$ | $A^1.formParameter = A.formParameter$ |

$$R6(P5): A.dimX = 2 * B^l.dimX + A^l.dimX$$

3.3 Topology

The geometric models of the examples given in the previous sections are topologically correct. As mentioned above we assume a 3D models to be topological incorrect if any two objects have other spatial relation than ‘disjoint’ or ‘meet’ (Egenhofer & Franzosa, 1991).

The semantic rules for the derivation of geometry are specified in a way that ensures a consistent topology. This is given by the monotony of the production rules and by the geometric shape of the represented objects. This can be seen by an inductive argument. Each symbol has an positive dimension in direction of growth

$$Flight.riser > 0, Flight.treadDepth > 0 . \quad (2)$$

For that reason the recursion generates a monotone translation. Since the dimension of any object is used as translation parameter the relation of any two neighbouring objects is meet and otherwise disjoint.

In order to prevent topological incorrectness we could also specify explicit constraints for topological correctness. The constraints base on the spatial relations meet and disjoint. Given a word W as a set of n symbols we claim the geometrical representation g of a any symbol to have the relation meet with its neighboured objects and disjoint with all other primitives:

$$\begin{aligned} & \text{meet}(g_i, m) \wedge \text{disjoint}(g_i, d) \\ & \forall g_i \in W, m \in M_i, d \in D_i, \text{ with } M_i \subseteq W, D_i = W \setminus M_i, i \in N, i \leq n \end{aligned} \quad (3)$$

Table 6 shows an excerpt of the according constraints implemented in semantic rules. The omitted rules are formulated analogously. Constraining the derivation in that way the correct topology of the model is ensured.

Table 6. Excerpt of topological constraints

| | |
|------------|----------------------------------|
| $R12(P4):$ | $meet(landing^1, Flight^1)$ |
| $R13(P4):$ | $disjoint(landing^1, landing^2)$ |
| $R14(P4):$ | $disjoint(landing^1, Flight^2)$ |
| $R15(P4):$ | $disjoint(landing^1, landing^3)$ |
| $R16(P4):$ | $meet(Flight^1, landing^1)$ |
| $R17(P4):$ | $meet(Flight^1, landing^2)$ |
| $R18(P4):$ | $disjoint(Flight^1, Flight^2)$ |

Since the production rules are topological correct already these additional constraints are redundant. Automatic identification of this redundancy as part of a post processing phase could remove this. Actually surveillance constraints and testifies the topological correctness of the remaining part. This affords capabilities of an automatic geometrical reasoning in 3D which however is beyond the scope of this paper.

4 ATTRIBUTE GRAMMARS IN A XML FRAMEWORK

We have implemented the application framework ‘XGep’ basing on the presented concepts. The framework consist of a XML schema defining the encoding of grammars in XML documents, a processing engine, and a graphical user interface (GUI). In the following we will present some details and results.

The XML schema defines the grammar as well as the derivation tree and the generated word i.e. the leaves of the derivation tree in depth first order. The schema supports three types of Chomsky grammar: regular, context-free, and context-sensitive. The processing part is implemented in Java to ensure cross platform ability. It is built modularly with open interfaces. So it

can easily be extended. In particular it uses an exchangeable ‘production decision mechanism’ which realises the method for choosing the production rule to be applied to a non-terminal symbol. This is particularly important if multiple rules derive the symbol. The default mechanism is a weighted selection. More complex methods could be implemented. The XGep graphical user interface supports the user in editing and processing a grammar in a comfortable way. It also provides tools for detailed exploring and step-by-step processing of the derivation tree. XGep’s main components are a grammar editor, the process control, a derivation tree explorer, and the export of words into visualisable 3D formats like X3D or OBJ. Other export formats are defined by filters which could be applied to XGep. Actually we implement a CityGML export filter.

The export to 3D graphic files corresponds to a geometric instantiation. The location and form attributes of each leaf node are used to transform a given geometric prototype. E.g. a riser (Tabs. 1,2) is represented by a rectangular shape which has the width *riser.width* and the height *riser.rise*. This rectangle has to be translated to the position *riser.refPoint*. Symbols can also be instantiated by volumetric prototypes or by a boundary representation of their surfaces.

A grammar for buildings of the Wilhelminian style quarter ‘Südstadt’ of the German city of Bonn has been implemented. Therefore the XGep framework has been used to specify and process the grammar. Figure 5 shows a result of this work.

More details, examples, and downloads can be found at <http://www.ikg.uni-bonn.de/xgep>.

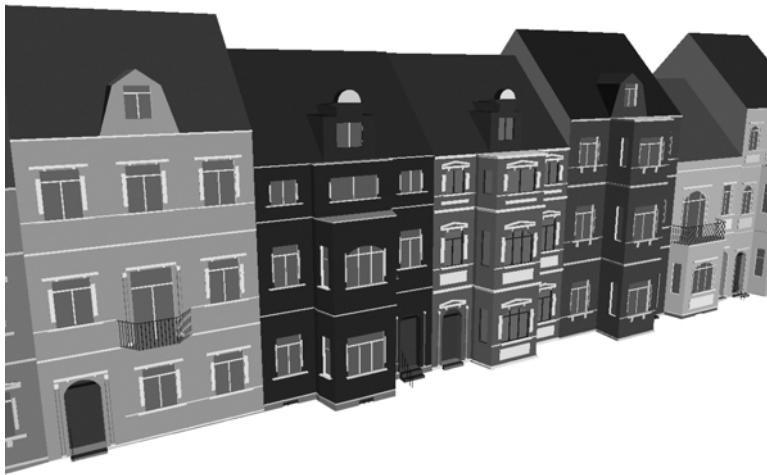


Figure 5. Screenshot of the 3D visualisation of randomly generated building models of the Wilhelminian style quarter ‘Südstadt’ in Bonn, Germany.

5 CONCLUSION AND FUTURE WORK

We have introduced a general concept for the usage of attribute grammars as a modelling language for 3D building models. The specification and constraining of aggregation and parameter inheritance by attributes and semantic rules has been shown. We have presented the usage of attribute grammars for the modelling and ensuring of mirror symmetry which is very common in architecture and thus in 3D city modelling. Furthermore we have given a concept for ensuring correct topology of the generated models. The implementation of the concept in the XGep framework has also been presented.

Future work should focus on the concept for ensuring correct topology anymore. The automatic validation of the topological correctness of the semantic rules is an open question. The implementation of the presented method into XGep is an actual task. We plan to connect the XGep grammar processing engine to a spatial enabled database management system (DBMS) like Oracle Spatial. This will offer the ability to store each terminal symbol as a spatial object into the DBMS and than use the DBMS for the verification of the spatial relation.

6 ACKNOWLEDGEMENTS

This work was done in the scope of the bilateral Sino-German bundle project ‘Interoperation of 3D Urban Geoinformation’ and the German bundle project ‘Abstraction of Geoinformation at the Multi-Scale Acquisition, Administration, Analysis and Visualisation’ both funded by the German research foundation DFG. The authors thank Martin Krückhans and Jan Behmann for their discussions and support on implementation.

REFERENCES

- Aho, Alfred V., Sethi, Ravi, & Ullmann, Jeffrey D. 1999. *Compilerbau Teil 1*. Oldenbourg.
- Alblas, Henk & Melichar, Borivoj 1991. Attribute Grammars, Applications and Systems, In Proceedings of International Summer School SAGA, Prague, Czechoslovakia, June 4-13, 1991, Volume 545 of Lecture Notes in Computer Science, Springer.
- Chomsky, Noam 1956. Three models for the description of language. *Information Theory, IEEE Transactions* 2(3):113–124.
- Chomsky, Noam 1959. On certain formal properties of grammars. *Information and Control* 2:137–167.
- DIN (Deutsches Institut für Normung e.V.) 2000. DIN 18065 - Gebäudetreppen.
- Dörschlag, Dirk, Gröger, Gerhard, & Plümer, L. 2008. Über die schrittweise Erstellung und Verfeinerung von Modellhypothesen für Gebäude (in German, translation of the title: About the stepwise generation and refinement of model assumptions for buildings). *Photogrammetrie, Fernerkundung, Geoinformation* (2008/3):157–164.
- Duarte, José Pinto 2002. Malagueira Grammar – towards a tool for customizing Alvaro Siza’s mass houses at Malagueira. Ph.D. thesis, MIT School of Architecture and Planning.
- Egenhofer, M. & Franzosa, R. 1991. Point-Set Topological Spatial Relations. *International Journal of Geographical Information Systems* 5(2):161-174.
- Fu, King Sun 1982. *Syntactic Pattern Recognition and Application*. Prentice-Hall.
- Geman, Stuart, Potter, Daniel F., & Chi, Zhiyi 2002. Composition systems. *Quarterly of Applied Mathematics* 60:707–736.
- Gröger, G., Benner, J., Dörschlag, D., Drees, R., Gruber, U., Leinemann, K. & Löwner, M. 2005. Das Interoperable 3D-Stadtmodell der SIG 3D (in German, translation of the title: The interoperable 3D city model of the SIG 3D). *Zeitschrift für Vermessungswesen* (130): 343-353
- Han, F. & Zhu, S. 2005. Bottom-up/top-down image parsing by attribute graph grammar. In Proceedings of Tenth IEEE International Conference on Computer Vision, 2005., Volume 2:1778-1785
- Knuth, Donald E. 1968. Semantics of context-free languages. *Theory of Computing Systems* 2(2):127–145.
- Knuth, Donald E. 1971. Top-down syntax analysis. *Acta Informatica* 1(2):79–110.
- Larive, Mathieu & Gaildrat, Veronique 2006. Wall grammar for building generation. In GRAPHITE ’06: Proceedings of the 4th international conference on Computer graphics and interactive techniques in Australasia and Southeast Asia, pages 429–437, New York, USA: ACM.
- Mäntylä, Martti 1988. *An Introduction to Solid Modeling. Principles of Computer Science*. Maryland, U.S.A. Computer Science Press.
- Müller, Pascal, Wonka, Peter, Haegler, Simon, Ulmer, Andreas, & Gool, Luc Van 2006. Procedural modeling of buildings. *ACM Transactions on Graphics* 25(3):614–623.
- Müller, Pascal, Zeng, Gang, Wonka, Peter, & Gool, Luc Van 2007. Image-based procedural modeling of facades. *ACM Transactions on Graphics* 26(3):85.
- OGC (Open Geospatial Consortium 2008). OpenGIS City Geography Markup Language (CityGML). Encoding Standard.
- Paakki, Jukka 1995. Attribute grammar paradigms - a high-level methodology in language implementation. *ACM Computing Surveys* 27(2):196–255.
- Ripperda, Nora & Brenner, Claus 2006. Reconstruction of facade structures using a formal grammar and rjmc. In Franke, K., Müller, K.-R., Nickolay, B., & Schäfer, R. (eds.), Proceedings of 8th DAGM Symposium, Berlin, Germany, September 12-14 2006. Volume 4174 of Lecture Notes in Computer Science, pages 750–759, Springer-Verlag.
- Russell, S. & Norvig, P. 2003. *Artificial Intelligence - A Modern Approach*. Pearson Education
- Schmittwilken, Jörg, Saatkamp, Jens, Förstner, Wolfgang, Kolbe, Thomas H., & Plümer, Lutz 2007. A semantic model of stairs in building collars. *Photogrammetrie, Fernerkundung, Geoinformation* (2007/6):415–427.
- Wonka, Peter, Wimmer, Michael, Sillion, Francois, & Ribarsky, William 2003. Instant architecture. *ACM Transactions on Graphics* 22(4):669–677.