Automatic Detailing of Parametric Sketches by Graph Transformation

S. Vilgertshofer and A. Borrmann

Chair of Computational Modeling and Simulation, Technische Universität München, Germany
E-mail: simon.vilgertshofer@tum.de, andre.borrmann@tum.de

ABSTRACT

In the scope of planning and realizing large infrastructural projects, it is reasonable to create product models as multi-scale models comprising multiple levels of detail (LoD). To avoid inconsistencies among the different LoDs, it is necessary to apply parametric modeling techniques which allow the automatic preservation of the model's consistency across the different LoDs in the case of modifications. Previous research in this area has revealed that the manual creation of consistency preserving parametric product models is a very complex, time consuming and error-prone task. Therefore, research concerning the automation of the detailing processes is necessary. This paper presents a detailing automation approach which is based on graph transformations. It discusses how two-dimensional parametric geometric models (sketches) can be represented by graphs and how detailing steps can be realized through graph transformation. A general approach to represent sketches by the use of graphs and the limitations applying to such an approach is described. It is discussed how geometric elements and corresponding parametric constraints of a sketch can be depicted by the nodes and edges of a graph and their attributes. Furthermore, the properties of the graph that are required for a non-ambiguous representation are analyzed. Based on those requirements a corresponding graph rewriting system is introduced. The functional capability of the presented theories were validated through a prototypic implementation executing the stepwise detailing of a sketch representing a shield tunnel section.

Keywords –

Level of Detail; Multi-scale Modeling; Graph Transformation; Parametric Modeling

1 Introduction

Carrying out the design and engineering of building projects successfully is a challenging process for all the parties involved. While even small and seemingly straightforward projects may evoke complex issues, this almost always applies to huge projects in which numerous boundary conditions and constraints as well as a vast number of experts from different domains are involved.

Obviously, it is the case for large infrastructure facilities like inner-city subway tracks that extend over long distances and need to be modeled and displayed at the right level of abstraction for an effective planning process. This implies the necessity to comprehensibly model project-wide overviews just as much as the lowest details for the particular user. It leads to widely differing scales and calls for the conception of plans and models utilizing multi-scale modeling methods.

Such methods are either not or only rudimentary implemented in BIM modeling tools, though already well established in the domain of geographic information systems by the use of multiple levels of detail (LoD).

To integrate this principle with BIM-methods, considerations concerning the display of a product model's geometry in varying scales but without any loss of consistency need to be taken into account.

A general method for the combination of semantic and geometric aspects of such a multi scale model is described in [1] and was used to develop a multi-scale product model for shield tunnels. It specifies a formal definition of five LoDs and investigates the (parametric) relations of geometric elements in different LoDs required for consistency preservation.

An essential part of such a product model is a geometric model depicting an object in multiple stages of a detailing process. The manual generation of such consistency preserving parametric multi-scale models is a very complex, time consuming and error-prone task. Analysis towards the automation of this process is the main motivation for the research outlined in this paper.

Therefore it presents a detailing automation approach which is based on graph transformation. In this context the paper discusses how two-dimensional parametric geometric models (sketches) can be represented by graphs and how detailing steps can be executed through the transformation of this graph.

To actually utilize this concept and to prove its functional capabilities the resulting graph data is used to automatically draw parametric sketches in a parametric CAD software tool. A sketch created thus is called the *evaluated sketch* in our approach.

In this scope the main focus of the research process is outlined by the following aspects:

1. Graph-based representation of parametric sketches.
2. Detailing of sketches by transformation of the representing graph.
3. Creation of the *evaluated sketch* via interpretation of the generated graph data.

The benefit of this approach lies in the possibilities offered by graph transformation techniques which include the definition of formal rules. Once defined, those rules may be executed multiple times to alter or detail sketches without the necessity of manually creating geometry or parametric constraints. Thereby, the user can focus on the actual design process since he does not need to concentrate on consistent parametric modeling. Instead, he only has to choose which rules to execute. The created sketches can be used as a basis for extrusion operations resulting in three-dimensional models.

To illustrate the overall approach, the detailing process of a simplified shield tunnel section and its corresponding graph-based representation is shown in Figure 1. To ensure readability of the figure, parametric constraints are neither displayed in the sketch nor labeled in the graph-based representation.

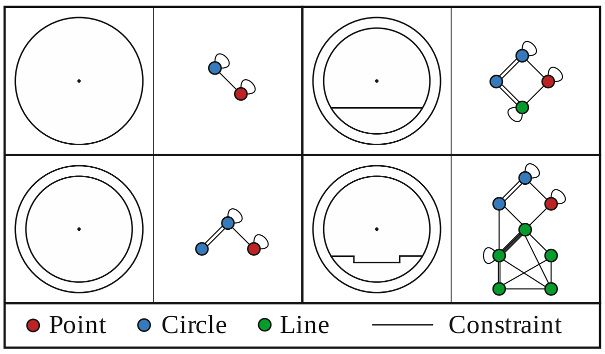

Figure 1. Detailing process of a sketch depicting a simplified shield tunnel section and its corresponding graph-based representation

The paper is organized as follows: Section 2 discusses related works and the theoretical background of the research. The general concept and issues of the graph-based approach are outlined in Section 3 while Section 4 describes the actual development results of a graph rewriting system. The paper concludes with a summary of the main findings and the discussion of future work.

2 Related Work and Theoretical Background

2.1 Integration of Multi-Scale Modeling in BIM

The idea of visualizing buildings or infrastructure facilities in different LoDs has been well established in the GIS domain for several years. For example, CityGML an XML-based data model for the representation of 3D city models comprises five LoDs [2]. Representations in coarse LoDs are generated from finer ones by abstraction. The design process inherent to construction projects on the other hand usually starts with a very general representation that gets refined stepwise throughout the planning phases [1].

Another aspect of the GIS approach is the lack of consistency preserving mechanisms in the data model - changes in one level of detail do not automatically lead to an update of the respective geometry in any other LoD [3]. This is caused by an independent storage of the geometry data in each LoD.

In [1], [3] and [4] the potential of this multi-scale approach is developed further and was successfully integrated into a comprehensive concept that permits the multi scale representations of building information models particularly directed to the modeling of shield tunnels. The challenge of a consistent representation throughout the multiple LoDs is addressed by the introduction of procedural modeling.

The ideas laid out in those publications are the main premise for the conceptual approach presented in the paper at hand. They describe the need for an automated detailing method to generate the geometry of the respective product models to supersede the disadvantageous manual generation of consistent parametric sketches or whole models.

2.2 Parametric Modeling and Formal Graph Theory

2.2.1 Parametric Modeling

Parametrical modeling techniques [5] are used in many CAD applications and provide the possibility not only to create drawings with definite dimensions but to define the rough layout of geometric objects by using constraints resulting in a parametrical sketch. The geometric elements such as points, lines, and circles of a sketch are not fixed in their location and dimension. Instead, they are arranged by geometric and dimensional constraints and thereby define a constraint problem [6].

This constraint problem is solved by the geometric constraint solver (GCS) integrated in the respective CAD application. The GCS analyses and solves the constraint problem. When more than one solution can be found (which is often the case) it proposes the solution most similar to the actual user input, suspecting the user to draw something close to his expected solution [7].

Since dimensions are not defined by actual values but by parameters which may refer to each other, quick alterations may be performed. Affected parameters are updated according to defined dependencies.

This permits the rapid simulation of various drafts or the adaption to modified boundary conditions without the effort of manually recreating the whole model.

To prove the concept shown in this research, we employed the parametric modeling software Autodesk Inventor.

2.2.2 Graph Rewriting

Graph rewriting is a mechanism to create a new graph out of an existing graph by altering, deleting or replacing parts (subgraphs) of the existing graph. The corresponding formal operation applied to the existing graph is called a graph rewrite rule $L \rightarrow R$. It is defined by a pattern graph L (also called left-hand side of the rule) and a replacement graph (right-hand side). When applied to an existing host graph, an occurrence of L is searched by pattern matching in the host graph. A found occurrence is then replaced by an instance of R. Depending on the definition of the rule, the search process also works for labeled and attributed graphs. A set of such graph rules is called a graph rewriting system.

There are several different approaches to graph rewriting, e.g. the Single-Pushout Approach (SPO) and the Double-Pushout Approach (DPO) which are both algebraic approaches [8]. Alternatively, the Node Label Controlled Mechanism [9] or the Hyperedge Replacement Mechanism may be used [8].

The graph rewriting software tool GRGEN.NET used in the scope of this research [10] is based on the SPO. It allows the definition of rewrite rules and their automatic execution on a given graph.

3 Conceptual Approach and Problem Analysis

3.1 The Graph-Based Approach

The expected advantage of using graphs for the representation of parametric sketches lies in the possibilities offered by the concept of graph rewriting. The execution of rewriting rules is supposed to automatically alter or detail a sketch represented by the particular graph.

It can be assumed that the presented graph-based approach to represent a geometric model whose topology is defined by parametric constraints is generally possible, because:

- Mathematically graphs are net-like structures used to formally represent objects and relationships between those objects. Due to their formal definition they permit consistent alterations to change the objects and various corresponding relations.
- Sketches may also be understood as structures composed of basic elements (points, lines, circles, etc.) which form a sketch through their positioning to each other. The geometric constraint solvers implemented in parametric modeling systems also use graphs to model the geometric elements and the parametric constraints composing a parameterized sketch [6].

3.2 Main Challenges

To allow insight into the research process leading to the findings presented in this paper an overview of the main problems inherent to the conceptual approach is given. Therefore, the issues listed in the introduction are discussed in detail. Their solutions are the basis for the properties of the developed graph rewriting system presented in Section 4.

3.2.1 Graph-Based Representation of Parametric Sketches

First of all, the problem of actually representing a parametric sketch has to be addressed.

The graph needs to be able to represent the geometric elements of a sketch as well as the parametric properties. As the topology of a sketch should be described mainly by the definition of parametric constraints, it is necessary to avoid absolute coordinates to achieve a full parametrization.

The constraint problem defined by the graph needs to be solved by the GCS of a CAD system as described in section 2.2.1. Unfortunately the CGS may produce more than one correct solution when no absolute coordinates are given. This may lead to an *evaluated sketch* which does not match the one desired by the user.

In this approach a particular graph-based representation must be interpretable unambiguously, though. It has to be definite and its interpretation must lead to only one *evaluated sketch*. Otherwise the creation of a sketch via graph-based data may lead to wrong or unintended results.

To satisfy both the avoidance of absolute coordinates and the unambiguous interpretation, the involvement of a GCS implemented in most parametrical modeling tools is necessary. The GCS can create a sketch by analyzing a set of geometric elements and corresponding parametric constraints to find a solution which meets all the requirements posed by the constraints. Unfortunately, many sets of parametrical constraints and geometric elements lead to various formally genuine solutions, although only one of those solutions matches the user's intention. Therefore the graph-based representation requires to comprise enough information for the sketch to be modeled accurately. This may lead to a necessity for coordinates describing the relative positioning of geometric elements to each other. This problem is further examined in Section 4.

The requirements of the graph-based representation is summarized as follows:

1. The graph must be able to represent geometric elements of different types.
2. The graph must be able to represent various parametric constraints.
3. The represented sketch needs to be fully parameterized and should include only the most necessary absolute coordinates.
4. For an interpretation of the graph by a GCS exactly one solution may exist.

3.2.2 Detailing of Sketches by Transformation of the Representing Graph

In general, the detailing of a sketch is to be understood as a process in which an initially primitive drawing is transformed into a more and more complex one by the alteration, replacement or addition of elements. This process is executed in separate steps.

To illustrate this process, the detailing of a sketch depicting the simplified cross-section of a shield-tunnel is shown in Figure 2.

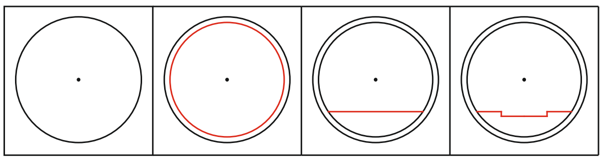

Figure 2. Detailing of a sketch depicting the simplified cross-section of a shield-tunnel

Due to the increase of complexity in the sketch, the graph representing the sketch has to become more and more complex accordingly (see also Figure 1). Therefore, methods of graph transformation need to be analyzed in respect to their capability of representing detailing steps. A corresponding graph rewriting system has to incorporate a set of rules formalizing those transformation methods. For any desired detailing operation in a particular sketch a particular rule has to be defined.

To achieve an automated computational execution of those rules they need to be implemented in a graph rewriting software tool. However, a successful implementation requires basic guidelines for the development of those rules. These guidelines must be followed to ensure that the application of any rule on the graph-based representation of any sketch will lead to a representation matching the requirements defined in Section 3.2.1.

In scope of the problem to detail a sketch by graph transformation the following questions can be summarized:

1. What are reasonable methods of graph transformation to represent detailing operations?
2. Which requirements need to be satisfied by the rewrite rules in order to produce only unambiguous representations?
3. What kind of information has a certain rewrite rule to comprise in order to allow the correct representation of a detailing operation?

Additionally the issues described in this section go hand in hand with the requirements listed in the previous section. Hence, every conclusion needs to be checked with regard to its conformity to all of the outlined conditions.

3.2.3 Creation of the Generated Sketch via Interpretation of the Representing Graph Data

As described in Section 1, the data modeled in the graph should be displayable in a parametric CAD application for further use. As the parametric constraints defined in the graph are also part of this model, its dimensions can be altered manually through the modification of the respective parameters in the CAD application. The implementation was carried out on the basis of the parametric modeling software Autodesk Inventor.

The graph-based representation is generated and transformed in a graph rewrite software tool (see Section 2.2.2). The interpretation of the resulting graph-based data is performed by a developed programm which accesses the API of Autodesk Inventor. It then executes the required construction operations to create and display the geometric elements and parametric constraints resulting in the display of the *evaluated sketch*.

While this workflow is generally applicable to any parametric CAD system, the process of interpreting the graph data needs to be adapted to the particular CAD system's API.

4 Development of a Graph rewriting system

To successfully develop a working graph rewriting system matching the pre-defined requirements, an iterative process was carried out. During this process, the general properties, the metamodel and the rewrite rules of the graph rewriting system were gradually adjusted until the results enabled an automatic detailing of sketches and their creation in the parametric modeling software Autodesk Inventor.

The main properties of the developed graph rewriting system, its metamodel and its rewrite rules are outlined in the following sections.

4.1 General

In this section, the general definitions of the graph and additional necessary properties are described.

The type of graph used in scope of this paper is a multidigraph or directed multigraph with loops (edges which start and end on the same node). It permits two edges to have the same start and end nodes. The multiple edges are necessary in case two of the represented geometric elements are linked with two or more parametric constraints. Loops are required for the representation of constraints that apply to only one geometric element.

As already depicted in Figure 1, nodes are generally used to represent the geometric objects of a sketch whereas edges are used for parametrical constraints.

4.1.1 Ports

Some geometric elements have multiple parts that a constraint may apply to. Lines for example have a starting point and an end point. For modeling a coincident constraint linking a line with a point, it is necessary to define which endpoint of the line it is referring to. To this end, so-called *ports* similar to an approach in [11] are assigned to geometric elements. They clearly define which part of the geometric element the constraint refers to. Depending on the particular type of geometric element, the amount of necessary ports varies.

Conceptually, a *port* is part of a geometric element. The actual *port* information is stored in the attributes of the edge representing the constraint attached to the object, though. As those edges always connect two nodes representing the linked elements, directed edges are essential to determine which *port* belongs to which element.

Figure 3 depicts the necessity for *ports* by illustrating two possible correct interpretations of a graph that does not contain any *port* information.

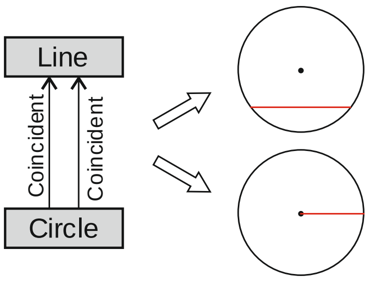

Figure 3. Interpretation of a graph without *port* information may result in multiple formally correct sketches

If the starting point of the line is defined to be coincident with the center of the circle and the endpoint is coincident with the arc of the circle, the interpretation of the shown graph will clearly result in the lower sketch.

4.1.2 Temporary Coordinates

If there are multiple solutions to a given constraint problem, the GCS chooses the solution which is most similar to an existing geometry. As we try to avoid absolute coordinates to create a parametric sketch, the GCS may produce a solution that is formally correct, but does not match the user's intention.

In order to match the requirement of an explicit representation, temporary coordinates are assigned to geometric elements. Those temporary coordinates form an approximation of the relative positioning of the geometric elements to each other. With their help, the interpretation of the graph-based representation by the GCS of a modeling software will always result in the intended sketch.

The temporary coordinates are stored in attributes of the nodes representing the geometric elements. Their values are calculated by the graph rewriting system during the graph transformation process. This causes the elements to be positioned at their temporary coordinates when created in the modeling software and thus performs an obvious solution for the GCS. The described method of prearranging the geometry does now lead to the intended solution.

4.2 Metamodel

A metamodel is a "data model that specifies one or more other data models" as described in ISO 11179. In terms of graph rewriting the metamodel therefore is an upfront definition of the entities (nodes and edges) a graph may consist of - including labels and attributes of those nodes and edges. A graph rewriting system needs such a definition comprising all the nodes and edges that may be used during any rewrite operation for successful execution of those rules [10].

The metamodel developed in scope of the presented research is shown in Figure 4. Its structure is based on an object-oriented approach and thereby allows the inherence of attributes. As nodes are used to represent geometric objects and edges are used for the representation of parametric constraints, the metamodel is structured accordingly.

The types of nodes and edges defined in the metamodel were developed in response to the requirements defined in Section 3.2.1. So far, they permit the representation of sketches consisting of *points*, *lines* and *circles*.

The types of parametric constraints that can be represented are divided in geometric and dimensional constraints. The supported geometric constraints represented by different types of edges are listed below:

- *Horizontal and Vertical*: A line is either parallel to the x- or y-axis of the coordinate system.
- *Fixed*: A geometric element is bound to remain on its assigned coordinates.
- *Collinear, Parallel and Perpendicular*: Those constraints always refer to a pair of lines. In case of the collinear constraints the lines lie on the same straight line. Parallel and perpendicular constraints work as implied by their names.
- *Concentric*: Two circles are forced to have the same center.
- *Equal*: Two lines or circles have the same length or radius.
- *Coincident*: Two geometric elements are coincident and are thus positioned at the same location in the sketch. If one or two of the elements are lines or circles, a port (see Section 4.1.1) has to be defined to explicitly determine which part (center, arc, starting point or endpoint of a line, etc.) of the geometric element is referenced. Therefore the representing edge needs to possess attributes defining the necessary ports.

Dimensional constraints describe distances, e.g. the distance between two points, the radius of a circle or the length of a line. The size of a dimensional constraint can either be a parameter or a numerical value. Parameters can be defined by mathematical functions and thereby refer to the parameters of other dimensional constraints.

Two types of edges were defined for the representation of dimensional constraints. To describe the length of a line or the radius of a circle, *DC_OneElement* is used, as only the dimension of one

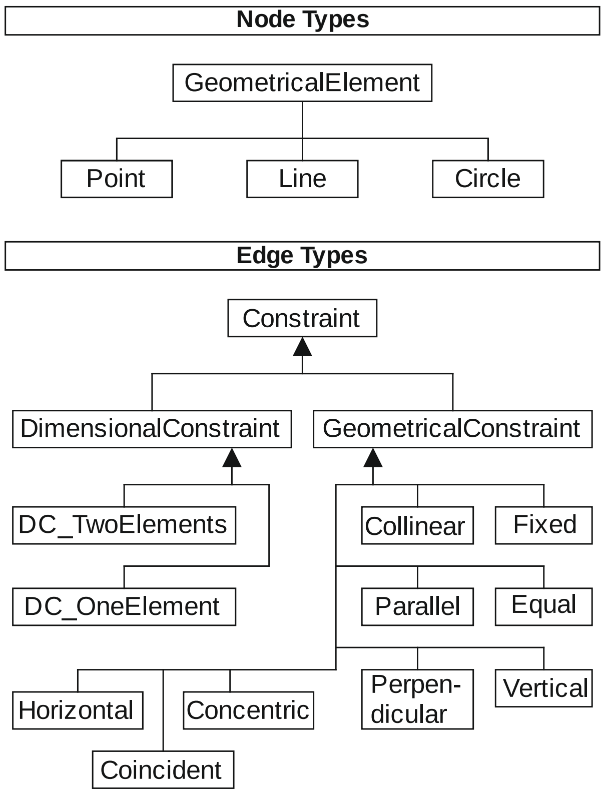

Figure 4. The metamodel of the developed graph rewriting system (without attributes).

geometric element is considered. If the distance between two elements needs to be defined *DC_TwoElements* is required. Analogously to the coincident constraint *ports* must be defined for lines or circles, to define which part of the geometric element is referenced.

Edges representing dimensional constraints need attributes to define their value or parameter (*parametric_value*). Additionally a parameter indicating if the constraint is driven is necessary. In case of *DC_TwoElements* two more attributes are required to describe the *ports*.

The metamodel developed in scope of this research is generally applicable to any sketch consisting of the listed geometric elements and parametrical constraints. It may be extended to allow the representation of additional elements or constraints.

4.3 Rewrite Rules

Rules are used to automatically transform the graph-based representation of a sketch into the representation of a refined or altered version of the sketch. Thereby the rule represents this particular detailing step or alteration.

Contrary to the metamodel the rewrite rules cannot be considered generally compatible to any type of sketch or detailing process. Hence, a rule has to be defined for each desired transformation operation individually.

In order to define a certain rule the initial sketch has to be compared to the desired modified sketch. The host graph can then be derived from the initial sketch. The desired result graph on the other hand needs to be constructed in a way that its interpretation will result in the modified sketch. By comparing the two graphs the applicable pattern and replacement graphs composing the rule can be determined. Comparison has to be performed with regard to the structure of nodes, edges and their attributes. This method to define a rule is illustrated in Figure 5.

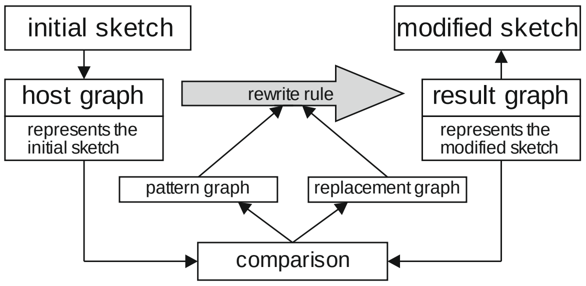

Figure 5. Flow-chart illustration of our approach to define a rewrite rule

To show the composition of a rule, the detailing operation shown in Figure 6 is analyzed.

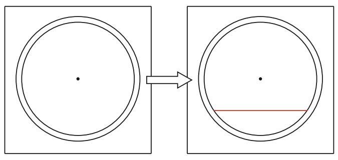

Figure 6. Exemplary detailing operation

The alteration is the addition of a line representing the floor of a shield-tunnel. Therefore a new node representing a line has to be added to the graph by connecting it with the appropriate edges.

The left side and the right side of the rewrite rule are illustrated in Figure 7.

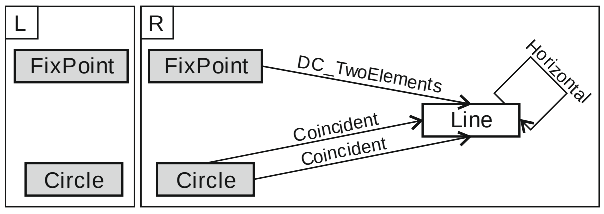

Figure 7. Rewrite rule that adds a horizontal line to a sketch. The endpoints of the line are defined to be coincident with the arc of an existing circle

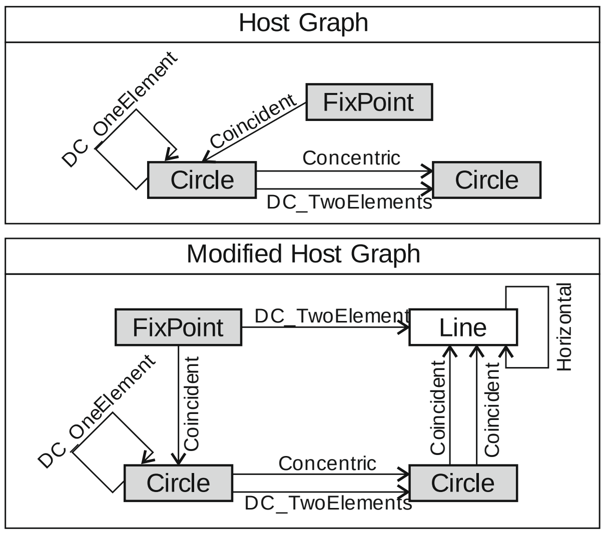

Figure 8. A graph before and after the execution of the rule shown in Figure 7

To demonstrate the effect of executing the rule the host and the result graph are shown in Figure 8. It should be noted that Figure 7 and 8 only contain labeled nodes and edges but do not show any attributes.

When executing a number of additional rewrite rules on a graph to generate a more detailed sketch, the graph gets more and more complicated and interconnected.

Figure 9 illustrates the outcome of a succession of multiple rewrite rules executed on a graph representing only a point on the tunnel alignment. Those rules generate the representation of a sketch depicting the shield tunnel section including the floor and two tracks.

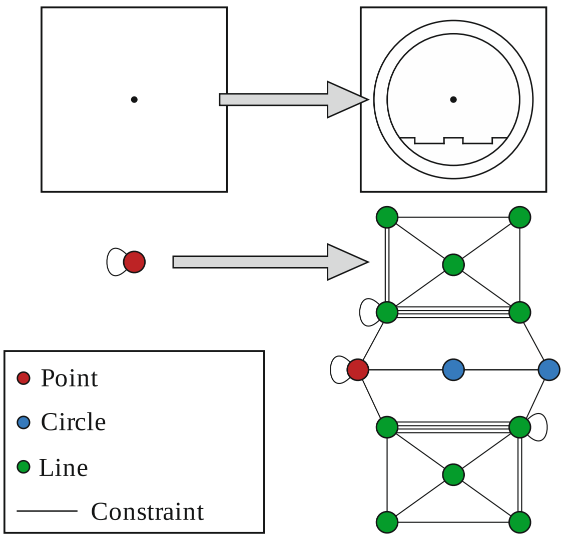

Figure 9. Comparison of two graphs representing a sketch in different stages of the detailing process

5 Conclusions and future work

This paper presents a concept for the graph-based representation of two-dimensional sketches and their automatic detailing by performing graph transformation operations using formal rules defined in a graph rewriting system.

Main contributions are the formal definition of the graph rewriting system composed of a largely universal metamodel and corresponding rewrite rules. A method to define those rewrite rules applying to specific detailing operations focusing on cross-sections of a shield tunnel is shown.

The functionality of this system has been validated through the development of a software prototype capable of creating sketches according to the represented parametric geometry in the parametric modeling system Autodesk Inventor.

Further research is focusing on extending the graph-based approach from the representation of two-dimensional sketches towards the modeling and detailing of three-dimensional tunnel sections and the actual possibility to represent multiple levels of details in a consistent manner, as described in [1], [3] and [4].

Acknowledgements

We gratefully acknowledge the support of the German Research Foundation (DFG) for funding the project under grant FOR 1546.

References

[1] Borrmann A., Kolbe T. H., Donaubauer A., Steuer H., Jubierre J. R. and Flurl M. Multi-scale geometric-semantic modeling of shield tunnels for GIS and BIM applications. *Computer-Aided Civil and Infrastructure Engineering*, DOI: 10.1111/mice.12090, 2014.

[2] Kolbe T. H. Representing and Exchanging 3D City Models with CityGML. In *Proc. of the 3rd International Workshop on 3D Geo-Information*, Seoul, Korea, 2008.

[3] Borrmann B., Ji Y. and Jubierre J. R. Multi-scale geometry in civil engineering models: Consistency preservation through procedural representations. In *Proc. of the 14th Int. Conf. on Computing in Civil and Building Engineering*, Moscow, Russia, 2012.

[4] Borrmann A., Flurl M., Jubierre J. R., Mundani R.-P. and Rank E. Synchronous collaborative tunnel design based on consistency-preserving multi-scale models. *Advanced Engineering Informatics*, 28(4):499-517, 2014.

[5] Shah J. J. and Mäntylä M. *Parametric and feature-based CAD/CAM: concepts, techniques, and applications.* John Wiley & Sons, New York 1995.

[6] Fudos I. and Hoffmann C. M. A graph-constructive approach to solving systems of geometric constraints. *ACM Transactions on Graphics (TOG)*, 16(2):179-216, 1997.

[7] Jubierre J. R. *Analysis and coupling of a Geometric Constraint Solver with a CAD application*. Master's Thesis, Technische Universität München. 2009

[8] Engelfriet J. and Rozenberg G. Graph grammars based on node rewriting: An introduction to NLC graph grammars. In *Graph Grammars and Their Application to Computer Science,* Springer Berlin Heidelberg, 1991.

[9] Rozenberg G. *Handbook of graph grammars and computing by graph transformation*, volume 1. World Scientific, Singapore, 1997.

[10] Geiß R., Batz G. V., Grund D., Hack, S. and Szalkowski A. GrGen: A fast SPO-based graph rewriting tool. In *Graph Transformations*. Springer, Berlin Heidelberg, 2006. S. 383-397.

[11] Helms B. *Object-Oriented Graph Grammars for Computational Design Synthesis*, Ph. D. thesis. Technische Universität München, 2013