Supporting feature-based parametric modeling by graph rewriting

S. Vilgertshofer[a] and A. Borrmann[a]

[a]Chair of Computational Modeling and Simulation, Technical University of Munich, Germany
E-mail: simon.vilgertshofer@tum.de, andre.borrmann@tum.de

Abstract –

Sophisticated geometric and semantic models are the basis for many applications in the field of Building Information Modeling. While the requirements in terms of detail, flexibility and conformity on those models and thus on the corresponding modeling tools increase, especially in the case of parametric and procedural modeling, open questions remain regarding the support of the user during the modeling process and the loss of modeling knowledge after finishing a modeling task. Graph Theory can be used when addressing these questions. It can be employed to represent parametric models in a vendor-neutral way and to capture modeling operations by formalizing them in graph rewrite rules. This paper describes the further development and generalization of graph-based model creation for the support of feature-based parametric modeling. We show how such procedural 3D models that are based on two-dimensional sketches can be represented by graphs and how modeling steps can be formalized by using rule-based graph rewriting. This approach enables a user to semi automatically reuse previously formalized modeling tasks, thereby supports and accelerates the modeling process and, additionally, allows the formal definition of expert engineering knowledge for later use and reapplication.

Keywords –

Graph rewriting; Parametric modeling; Modeling support

1 Introduction

Realizing the design and engineering of construction projects successfully is a challenging process for all the parties involved.

While even small and straightforward projects may evoke complex issues, this is typically the case for large projects in which various boundary conditions and constraints as well as a vast number of participants from different areas of expertise are involved.

The technological advancements developed alongside the ongoing introduction of Building information modeling have addressed those challenges and support designers and engineers in their daily work and their interdisciplinary communication. However, the availability of sophisticated models containing geometry as well as semantics are a major requirement for many applications and workflows. Use scenarios such as automated construction progress monitoring [1], automated code compliance checking [2], automated cost analysis or BIM-based generation of construction schedules [3] would not be possible without underlying BIM-models comprising various kinds of information.

The task of creating models is therefore an indispensable prerequisite for the use cases mentioned above as well as a large variety of additional scenarios. The modeling of shield-tunnels, for example, can benefit from geometric models comprising different Levels of Detail (LoD) as introduced by Borrmann et al. [4]. 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 case of alterations. The research by Borrmann et al. has revealed that the manual creation of consistency preserving parametric product models is a very complex, time consuming and error-prone task. An approach by Vilgertshofer and Borrmann [5,6] introduces the possibilities of using graphs and graph transformation to support the necessary modeling process by formalizing parametric 2D sketching operations and subsequent procedural operations, which semi-automatically create 3D models for linear parts of shield-tunnel facility models. In this context, "linear" denotes the part of the model that rely solely on the course of the alignment, the most important basis for infrastructure facilities ranging over longer distances. Parts of the model which are positioned at specific points on the alignment (crosscuts, fire exit shafts, etc.) are called "non-linear".

An overview of the concept of this existing approach is shown in Figure 1, which conceptually illustrates how a model is represented by a graph and how this graph is

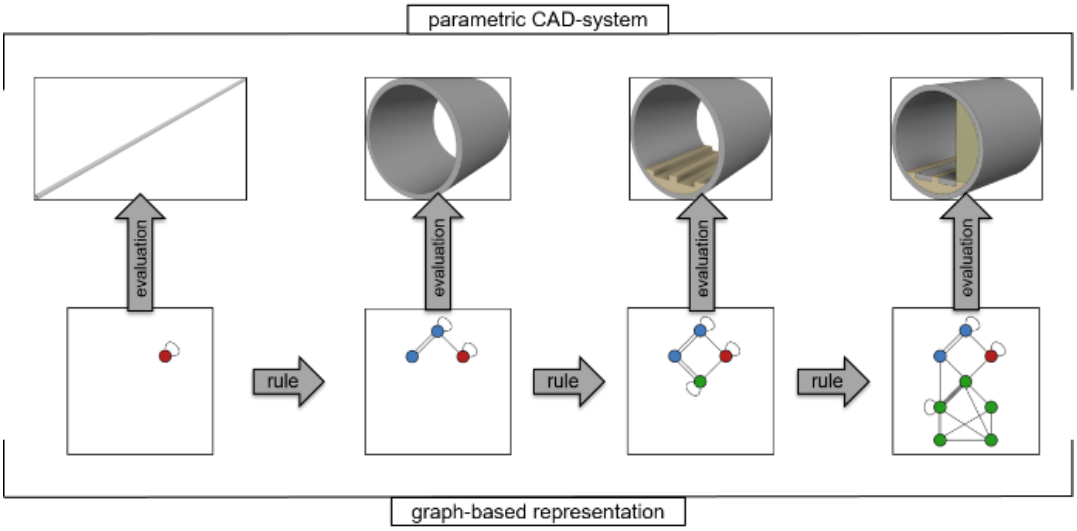

Figure 1: Conceptual illustration of using graph rewrite rules that formalize detailing steps of the graph-based representation of a shield-tunnel model. Each instance of the graph can be evaluated to create an actual model in a parametric CAD-system.

transformed by applying predefined rewrite rules. An arrow "rule" indicates the application of a rewrite rule. At each stage of this process, a parametric CAD system can interpret and process the graph in order to create an editable parametric model. This model is further called the evaluated model. An arrow labeled "evaluation" marks the generation of the evaluated model out of the graph. Note, that the graphs in Fig. 1 do not actually represent the respective geometry. They merely illustrate the presented concept.

In this paper, we will show how this approach can be refined and developed further in order to enable the representation and formal generation of more complex models. In the case of shield-tunnel models we extend the method towards the additional support of nonlinear (meaning "not based solely on the alignment") geometry parts, such as fire exit shafts or crosscuts. Furthermore, we will introduce a possibility to use the graph-based method for the semi-automatic parametric model generation in the scope of high-rise construction, namely steel connections (Figure 2). To achieve this, we will show how we can refine our graph rewrite system to allow the representation of procedural models comprising assemblies and constraints on assembly level.

Figure 2: Detailing of a steel connection.

The paper is structured as follows: Subsequent to the introduction, Section 2 will give an overview of previous and related research as well as the theoretic background of parametric modeling and graph rewriting. Section 3 will introduce our method of graph-based description of parametric modeling and show how this method is further refined. Additionally, it will summarize our implementation work and possible use cases of our extended method. The paper concludes with a summary.

2 Background and related work

The benefits of the computer-aided or (semi-)automated creation of designs and models have been addressed by researchers before. This section puts the presented approach in context of a short overview of existing approaches. It further presents the theoretical background of the proposed methodology in terms of parametric and procedural modeling and graph rewriting.

2.1 Modeling support

Computers successfully support the process of generating technical drawings or product models and parametric CAD software is widely used and enormously valuable in the building sector [7]. However, the main purpose of such software is to assist an engineer in his creative design work, which is one of the most complex human tasks, as it depends on the consideration of various constraints to obtain satisfactory solutions [8]. Therefore, a further step is the development of methods and tools, which actively support a designer by automatically generating whole sets of design variants or

by the automation of repetitive and trivial tasks in the design process. As the concept presented in this paper contributes to this field of research, major approaches, which also utilize graph representations, are summarized here.

In the field of Computational Design Synthesis (CDS), Helms [9] uses a graph grammar for the computational synthesis of product architectures. Design knowledge is captured in a port-based metamodel and the procedural design rules of the grammar. Hoisl [10] presents an approach for creating a general spatial grammar system that introduces interactive definition and application of grammar rules in the scope of CDS. It aims at actively supporting a designer in the modeling process using mechanical CAD systems. The approach by Kniemeyer [11] in the domain of biology makes use of a graph grammar to design and implement a language to support the functional-structural modeling of plants.

Furthermore, Lee et al. [12] fundamentally describe how parametric building object behavior can be specified for building information modeling systems. Their approach shows how a common method to "describe the design intent in order to share and reuse the user-defined parametric objects" between collaborating experts can be realized. This approach, however, is not intended to automate the encoding of parametric object behavior definitions.

2.2 Parametric modeling

The concept of parametric, procedural and feature-based modeling was developed in the 1990s [13] and is by now well established and used in many commercial and open source CAD applications such as Autodesk Inventor, Siemens NX and FreeCAD.

While a pure geometric model stores only the coordinates of the geometric elements, the concept of parametric feature-based modeling is to store the sequence of sketching and subsequent 3D modeling operations: The construction history of the model. Generally, a construction history has the following structure: Parametric geometric 2D models (sketches) are composed of geometric objects and parametrical constraints. During the creation of a sketch in a parametric CAD application, a system of constraints and objects is defined and forms a constraint problem. A geometric constraint solver (GCS) [14] can solve such a problem. Schultz et al. [15] define the set of parametric constraints that is implemented by all major constraint solvers as the standard geometric constraint language. It comprises the dimensional constraints for distances and angles as well as the following geometric constraints: *coincident, collinear, tangential, horizontal, vertical, parallel, perpendicular and fixed*.

A parametric sketch created in this manner can then be used as the basis for an extrusion, sweep, loft or rotation to create a 3D object, a so-called *feature*. By applying Boolean operations, several of these features are then combined to models that are more complicated and result in parts. The combination of various parts lead to the creation of an *assembly*. On assembly level, different parts are arranged by *mating conditions*, which are basically complex parametric constraints applied to points, lines or surfaces of parts.

The main advantage of a 3D model created in this manner is, that it allows changes of any operation in the construction history without losing the consecutive modeling operations. Therefore, alterations are easier, and errors can be fixed without the necessity of a complete remodeling. This modeling technique, however, relies on a deep understanding of its basics and therefore requires extensive training of possible users, as a multitude of constraints and parts lead to very complicated models that can get almost unmanaged without knowing the originator's intentions Lee et al. [12]. Our approach therefor aims at introducing automation mechanisms into parametric feature-based modeling.

2.3 Graph rewriting

The presented approach for automating the detailing process in this paper is based on graph theory and also uses graph rewriting methodology as comprehensively described by Rozenberg et al. [16]. We employ graphs and graph rewriting mechanisms to enable the representation and the modification of procedural parametric models. An application of graph rewriting to semi-automatically create and alter parametric sketches has been presented in Vilgertshofer and Borrmann [5] and was further developed [6].

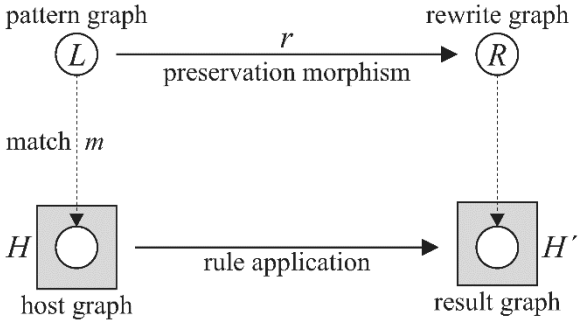

Figure 2: Graph rewriting via the SPO (inspired by Blomer et al. [19]).

Graph rewriting operations are used to create a new graph out of an existing graph by altering, deleting or replacing parts (subgraphs) of the existing graph. The changes are formalized through graph rewrite rules

written as $L \rightarrow R$. A graph rewrite rule is defined by a pattern graph L and a replacement graph R. When a rule is applied to a graph (called the host graph), this graph is searched to find a subgraph that matches the graph pattern defined by L. A successful matching leads to the replacement of L with R under the consideration of a preservation morphism r. This preservation morphism controls how R substitutes or alters an instance of L in the host graph. The outcome of this rule application is called the result graph H' as illustrated in Fig. 2.

There are several different approaches to graph rewriting. Two main examples are the Single-Pushout Approach (SPO) and the Double-Pushout Approach (DPO) [17].

3 Conceptual approach

This section will give an overview of the basic concept of our existing approach and show how it was extended

3.1 Graph-based description of parametric models

In the following section, we will first give a short comparison of our work so far and discuss its limits. Thereafter, we describe how we revise and extend it to generate a wider range of parametric models.

3.1.1 Overview

Parametric modeling CAD-systems use geometric elements such as points, lines or circles as primitive planar entities. The parametric constraints as described in Section 2.2 define the topology of these entities and result in parametric sketches. Those are the basis for further procedural modeling operations, which create 3D features. To represent a procedural parametric model by means of a graph, it is necessary to define, which types of geometric elements, parametric constraints and procedural modeling operations are employed. In the scope of our research so far, we generally considered the following types:

- geometric elements: *point, line, spline circle, arc*
- parametric constraints:
  - geometric constraints: *coincident, collinear, equal, concentric, horizontal, vertical, parallel, perpendicular, fixed*
  - dimensional constraints: *dimensions of one geometric element, distances between two geometric elements*
- procedural modeling operations: *workplane, extrusion, sweep*

This listing roughly reflects the standard geometric constraint language defined in Schultz et al. [15] summarizing the most common operations provided by any parametric CAD system (see Section 2.2).

3.1.2 Formalization

For formal representation of these items, a graph metamodel describes the necessary attributed types of graph nodes and edges are. They can then be instantiated during the generation of a graph. The metamodel also forms the basis for the definition of graph rewrite rules, which formally describe modeling steps. An end user can apply those instead of manually executing the underlying procedural or parametric modeling operations. Fig. 1 in Section 1 conceptually illustrates how a graph represents a model and how the application of predefined rewrite rules transforms this graph.

The graph representing a procedural geometry model is a directed multigraph with loops $G = (V, E, T^v, T^e, s, t, lb, ty^v, ty^e, att)$. It is defined as follows:

- $V = V_P \vee V_S$ is a nonempty finite set of vertices. Elements of V_P are vertices that represent procedural modeling operations, while elements of V_S represent geometric objects in a sketch.
- $E = E_P \vee E_S$ is a nonempty finite set of edges. Elements of E_P are used to represent general relations or dependencies between the procedural operations and allocate geometric elements to a specific sketch. Elements of E_S represent parametric constraints of a geometric element or between two geometric elements.
- $V_P \wedge V_S = \emptyset$ and $E_P \wedge E_S = \emptyset$.
- $s : E \rightarrow V$ is a mapping that indicates the source node of all edges.
- $t : E \rightarrow V$ is a mapping that indicates the target node of all edges.
- Σ is an alphabet of labels of vertices and edges.
- $lb : E \vee V \rightarrow \Sigma$ is a labeling function.
- $T^v = T_P^v \vee T_S^v$ is a set of types for the vertices in V. T_P^v and T_S^v are sets of types for nodes in V_P and V_S respectively.
- $T^e = T_P^e \vee T_S^e$ is a set of types for the edges. T_P^e and T_S^e are sets of types for nodes in E_P and E_S respectively.
- $ty^v : V \rightarrow T^v$ is a typing function for the vertices, such that $ty^v(V_P) \wedge ty^v(V_S) = \emptyset$.
- $ty^e : E \rightarrow T^e$ is a typing function for the edges, such that $ty^e(E_P) \wedge ty^e(E_S) = \emptyset$.
- At is a set of attributes of vertices and edges.
- $att : E \vee V \rightarrow At$ is an attributing function.

The metamodel describes the possible set of types T^v and T^e of the graph entities V and E. They can then be instantiated to execute a rewrite rule to create or alter the graph. Additionally, the metamodel defines the attributes of a certain type as well as conditions that control which nodes and edges may be incident or which node types can

be adjacent. As the type of a graph entity clearly determines which attributes that entity has, an attributing function is not given.

Instead of using separate graphs to represent sketches and subsequent procedural operations we concluded that combing all information needed for the representation of a particular model should be embedded in a single graph. This is realized by integrating graphs that represent a sketch into the procedural graph. The benefits of this method are presented in [6] in detail. We still conceptually separate the subgraphs representing sketches and the procedural operations that subsequently create 3D features and therefore use the terms *sketch graph* and *procedural graph*.

3.1.3 Limits

While the presented method enables the automated generation of basic shield tunnel models, based on the alignment, limitations occur. This is especially the case, when we approach the question of non-linear geometry or use cases in other domains. Creating and placing various features in a model proves quite difficult when there is no possibility to arrange them without altering the position of the sketches. In the parametric feature-based modeling theory, so-called assemblies remedy this problem. One or more features are combined into one part, whereas an assembly consists of multiple parts. While each part has its own local coordinate system to position one or more features (each sketch has its own local coordinate system, too), the assembly itself defines yet one more coordinate system in which the different parts are positioned. This placement is achieved by using either fixed coordinates (which is usually the case for the first part to be positioned) or by placing one part relatively to another one. The relative positioning works quite similar to the parametric constraints, which define the topology of the geometric elements in a sketch. In this context, however, the term mating conditions is used. Basic mating conditions define points, lines or faces of a part to be constrained to those of another part in terms of being coincident (points), collinear (lines) or on the same plane (faces).

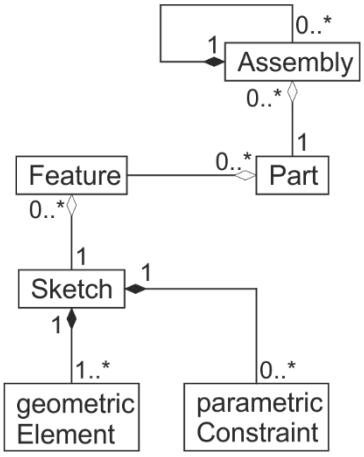

Figure 4: Structure of a model consisting of sketches, features, parts and assemblies.

In this regard, we also encountered the problem of referencing geometric entities that are the result of a

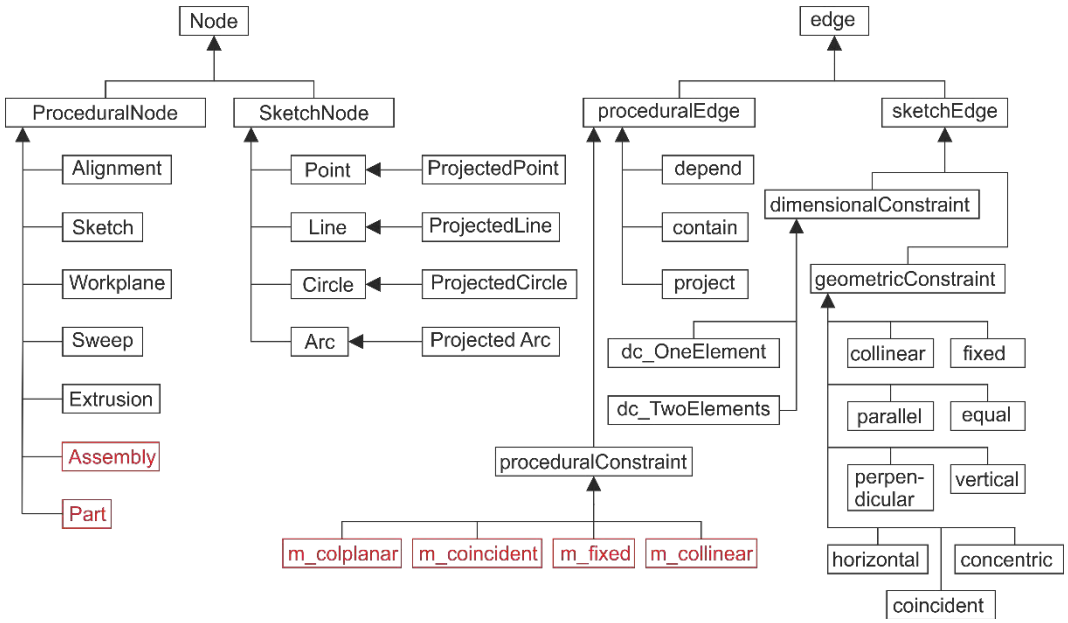

Figure 5: Extended version of the graph metamodel.

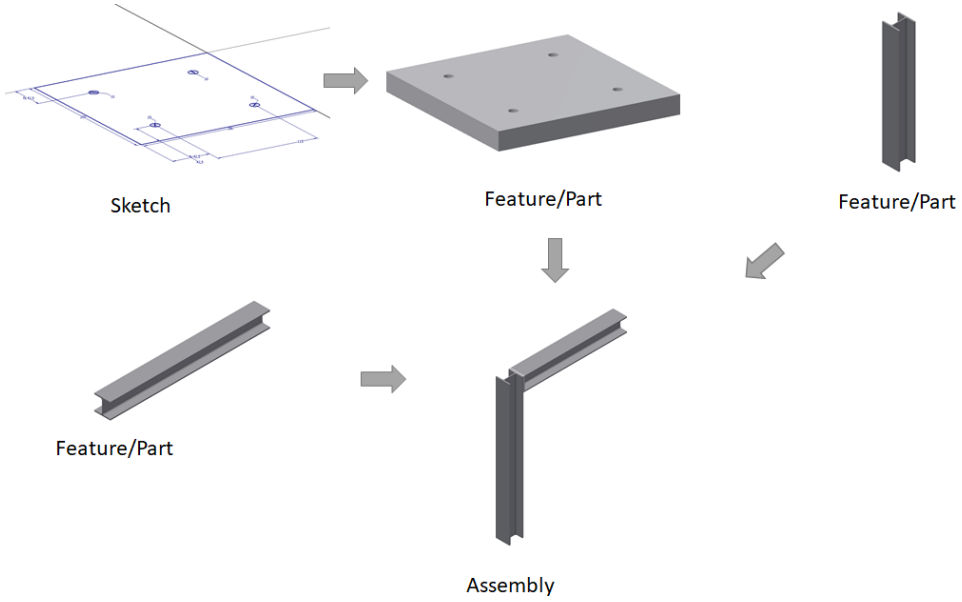

Figure 6: Modeling of a steel connection with the use of features/parts as an assembly.

procedural operation in the graph. The extrusion of a rectangle, for example, forms a cuboid. Here, the four points and four lines that the sketch describing the rectangle contains, are represented in the graph. Another eight lines, however, bound the resulting cuboid. Additionally, four new points connect those lines and the cuboid itself comprises six faces. However, the graph does not represent these 18 new entities and it is therefore not possible to reference them in subsequent graph transformation operations. Therefore, we need to refine the rules creating subgraphs that represent feature objects to comprise such geometric entities, if they objects need to be referenced by consecutive rewriting operations.

The necessary extensions to our graph metamodel in order to include assemblies and mating conditions are described in the following subsection.

3.2 Extension of the approach

As the graph metamodel is the formal description of types of graph nodes and edges that can be instantiated, it has to be extended in order to cover the representation of the described modeling operations on an assembly level. Figure 5 depicts the previous version of the metamodel in black color, while the necessary extensions are drawn in red color.

Most important extensions are the new node types *part* and *assembly*. They are used in a similar manner as sketch nodes are used to group the geometric elements of a sketch: Part nodes group one or more features created from sketches, whereas assembly nodes group one or more parts. Furthermore, constraints in the procedural context are added, to define the relative positioning of the parts in an assembly.

While this refinement is not extensive in terms of new node and edge types, it allows us to model more complex geometry than in the previously presented approach. We are now able to construct a model consisting of more than one part and to position these parts relative to one another. This decomposition of the model allows us to generate subgraphs representing model parts that are either completely independent from one another or only related by mating constraints.

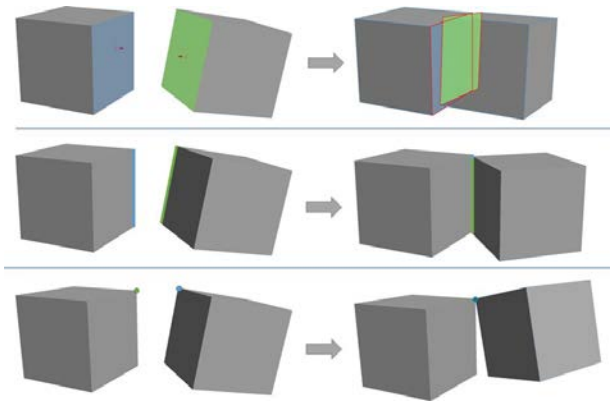

Figure 7: A model before and after applying three different types of mating constraints: coplanar, collinear and coincident.

These mating constraints are much better suited to the relative positioning of three-dimensional objects to one another than parametric constraints, as we do not have to consider the positioning of workplanes or the projection

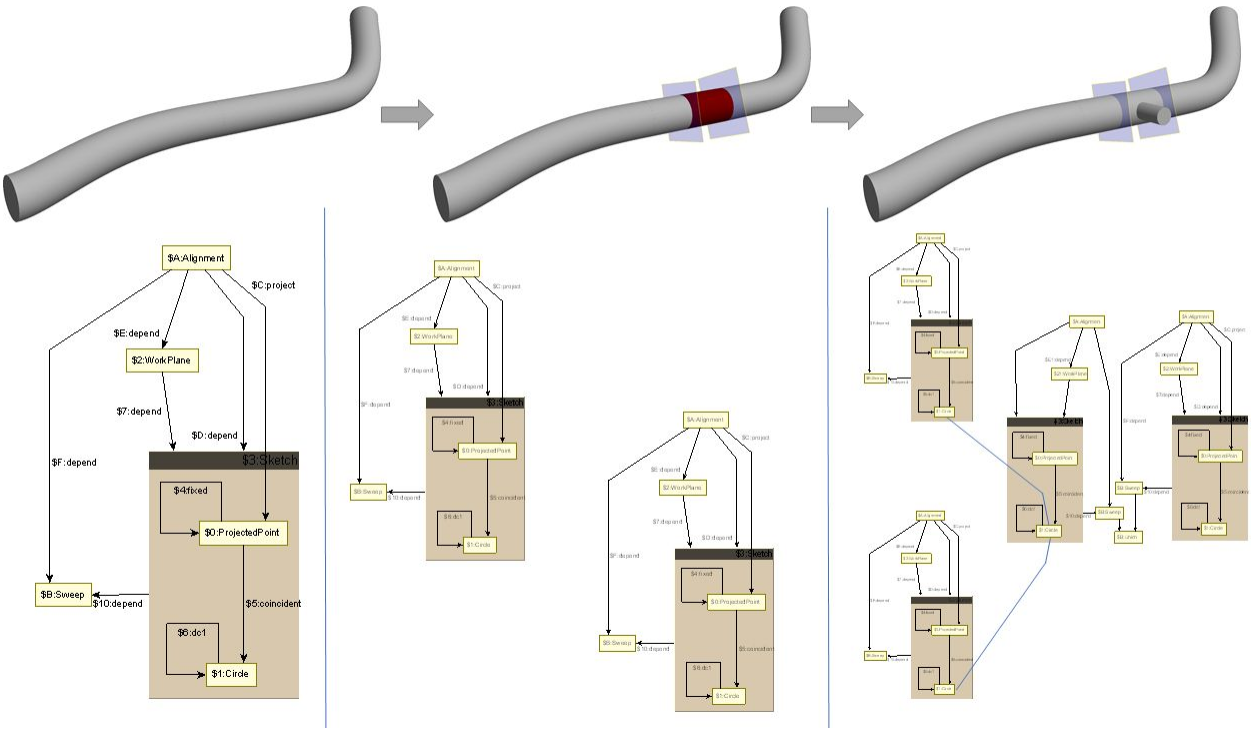

Figure 8: Process of adding a crosscut to a tunnel model. Bottom: States of the graph representing the model. Top: Geometric result of the graph transformation process.

of existing geometry between sketches. During the modeling of parts those operations are still necessary and helpful, though. When positioning parts in an assembly they can also be rotated via the application of mating conditions. As this is not the case with features combined in a part without using a procedural operation, this also gives us more freedom in the positioning process. The graph is in its definition now much more conform to the general concept of parametric feature based modeling.

As described before, the entities that mating constraints link to one another have to be present in the graph in order to apply those constraints. We concentrated on two possible solutions for this problem: We can either define a rewrite rule in a way that all new entities (points, lines and faces) are also created within the representing graph and can be referenced by further rewrite rules. Another possibility is to only create those entities that are necessary for later rewriting operations. This, of course, is only reasonable if we know at that point, which entities this will be. We are still considering which of these solutions is more constructive or if they should both be implemented simultaneously.

3.3 Use cases and implementation

For the definition and extension of the graph rewrite system consisting of a metamodel and appropriate graph rewrite rules, the graph rewrite generator GrGen.NET [18] has been used while the generation of the evaluated sketch is performed with the commercial parametric CAD application Autodesk Inventor. Inventor contains a geometrical constraint solver, which interprets the constraint problem defined by the graph. A software prototype was developed to utilize both the functionalities of GrGen.NET and of Autodesk Inventor to apply rewrite rules and perform the consecutive creation of the evaluated model.

In order to verify the improvements made to the graph rewrite system we examined two test scenarios. First, we employed the graph system to model the connection of two steel beams (Figure 6). Here, the connecting plate is designed from extruding a sketch to create a 3D feature. A part consisting only of this feature is then combined with two other predefined parts (the beams) in an assembly. To position those three parts in accordance with each other, several planes of the respective parts were constrained by mating conditions.

In the scope of modeling non-linear geometry of shield-tunnels, a crosscut was added to an existing model of a tunnel. We also realized this by using the introduced assembly nodes. While the tunnel model without the crosscut would be modeled as only one part, we now cut the alignment at both sides of the future position of the crosscut. As we now have three alignment sections, we use them as basis for three different parts. The two outer parts are created by reapplying the existing rules that

rewrite the graph to create the linear 3D geometry. The inner part however is represented by a new subgraph. This subgraph is created by executing a corresponding rewrite rule that creates the representation of a tunnel section comprising the opening for the crosscut. This process is graphically illustrated in Figure 8. Thereby we create three rather independent subgraphs representing the three model parts shown in the figure. Mating constraints are then used to combine them to a consistent model. In the graph representation the subpgraphs are therefore connected by edges representing those mating constraints. For example, they are used to mate the faces of two tunnel parts that have to align in order to keep those parts in position respective to each other.

4 Summary

The presented research introduces and extends a concept for the graph-based representation of product models and their automatic generation and detailing by performing graph rewrite operations based on formal rules defined in a graph rewriting system.

It has already been successfully applied to the product models of shield-tunnels and the automatic creation of consistency preserving multi-scale versions of such models. The main contribution is the further elaboration of the underlying graph rewriting system that enables the generation of more complex graphs covering the representing of a larger variety of parametric representations. To prove the feasibility of our approach, the graph rewriting system has been implemented in the graph rewriting tool GRGEN.NET.

Further research will focus on creating a larger set of rewrite rules, which enables end users to create more diversified models in different contexts.

Acknowledgements

We gratefully acknowledge the support of the German Research Foundation (DFG) for funding the project under grant FOR 1546. We also want to thank the members of the 3DTracks research group for their support and the productive discussions.

References

[1] Braun A., Tuttas S., Borrmann A. and Stilla, U. A concept for automated construction progress monitoring using BIM-based geometric constraints and photogrammetric point clouds. *ITcon*, 20 (8), pp. 68-79, 2015

[2] Preidel C., and Borrmann A. Towards code compliance checking on the basis of a visual programming language. *ITcon* 21. 2016

[3] Sigalov K. and König M. Recognition of process patterns for BIM-based construction schedules. *Advanced Engineering Informatics*, 2017.

[4] 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 Eng.* 30 (4), pp. 263-281, 2015.

[5] Vilgertshofer S. and Borrmann A. Automatic Detailing of Parametric Sketches by Graph Transformation. *Proc. of the 32nd ISARC*, Oulu, Finland, 2015

[6] Vilgertshofer S. and Borrmann A. Using graph rewriting methods for the semi-automatic generation of parametric infrastructure models. *Advanced Engineering Informatics*, 2017

[7] Camba J.D. and Contero M. Parametric CAD modeling: an analysis of strategies for design reusability. *Computer-Aided Design*, 74. 2016.

[8] Bhatt M., Borrmann A., Amor R. and Beetz J. Architecture, computing, and design assistance. *Automation in. Construction*, 32, 2013.

[9] Helms B. and Shea K. Computational synthesis of product architectures based on object-oriented graph grammars, *J. of Mech. Design*, 134, 2012.

[10] Hoisl F.R. Visual, Interactive 3D Spatial Grammars in CAD for Computational Design Synthesis, *Ph.D. thesis*, TU München, 2012.

[11] Kniemeyer O. Design and Implementation of a Graph Grammar Based Language for Functional-Structural Plant Modelling. *Ph.D. thesis*, BTU Cottbus, 2008.

[12] Lee G. Sacks R., and Eastman C.M. Specifying parametric building object behavior (BOB) for a building information modeling system. *Automation in Construction*, 2006.

[13] Shah J.J. and Mäntylä M. Parametric and Feature-Based CAD/CAM: Concepts, Techniques and Applications, 1995.

[14] Fudos I. and Hoffmann C.M. A graph-constructive approach to solving systems of geometric constraints. *ACM Trans Graph*, 16, 1997.

[15] Schultz C., Bhatt M. and Borrmann A. Bridging qualitative spatial constraints and feature-based parametric modelling: expressing visibility and movement constraints. *Adv. Eng. Inf.* 31, 2017

[16] Rozenberg G. Handbook of Graph Grammars and Computing by Graph Transformation, vol. 1, World Scientific, 1997.

[17] Heckel R. Graph transformation in a nutshell. *Electron. Notes Theoret. Comp. Sci*. 148, 2006.

[18] Geiß R., Batz G.V., Grund D., Hack S. and Szalkowski A. GRGEN: A fast SPO-based graph rewriting tool. *Proc. ICGT* 4178, 2006.

[19] Blomer J., Geiß R. and Jakumeit E. The GrGen.NET User Manual, 2014.