Graph Transformations for Modeling *hp*-Adaptive Finite Element Method with Mixed Triangular and Rectangular Elements

Anna Paszyńska¹, Maciej Paszyński², and Ewa Grabska¹

¹Faculty of Physics, Astronomy and Applied Computer Science,
Jagiellonian University, ul. Reymonta 4, 30-059 Cracow Poland
anna.paszynska@uj.edu.pl

²Department of Computer Science,
AGH University of Science and Technology,
Al. Mickiewicza 30, 30-059 Cracow, Poland

Abstract. The paper presents composition graph (CP-graph) grammar, which consists of a set of CP-graph transformations, suitable for modeling transformations of two dimensional meshes with rectangular elements mixed with triangular elements. The mixed meshes are utilized by the self-adaptive *hp* Finite Element Method (FEM) extended to support triangular and rectangular elements. The *hp*-FEM generates a sequence of mixed triangular and rectangular element meshes providing exponential convergence of the numerical error with respect to the mesh size. This is done by executing several *h* or *p* refinements over an initial mesh. The mixed finite element mesh is represented by attributed CP-graph. The proposed graph transformations model the initial mesh generation as well as mesh refinements. The proposed extended graph grammar has been defined and verified by using implemented software.

1 Introduction

The topological structure of the finite element mesh, with a hierarchy of vertices, edges and faces has been proposed by [1] to support mesh generation and data storage. The first attempt to model mesh transformations by applying the graph grammar concept has been proposed by [2] for the regular triangular two dimensional meshes with *h* adaptation. This has been done by using the quasi context sensitive graph grammar. However, the applicability of the quasi context sensitive graph grammar seems to be limited, since the mesh transformations utilized by adaptive algorithms are context dependable. The Composite Programmable graph grammar (CP-graph grammar) has been introduced by [3,4,5] as a tool for a formal description of various design processes. The CP-graph grammar expresses a design process by means of the graph transformations (called productions) executed on the CP-graph representation of designed objects. In this paper the CP-graph grammar is used to model mesh transformations executed by the 2D self-adaptive *hp* Finite Element Method (FEM) with triangular and rectangular finite elements [8]. The paper is an extension

of the CP-graph grammar model introduced in [6] for rectangular finite element meshes, and the CP-graph grammar model introduced in [7] for triangular finite element meshes. The mixed triangular and rectangular element meshes are represented by attributed CP-graphs. The graph grammar consists of a set of graph transformations, called *productions*. Each production replaces a sub-graph defined on its left-hand-side into a new sub-graph defined on its right-hand-side. The left-hand-side and right-hand-side sub-graphs have the same number of free in/out bounds, and the corresponding bounds are denoted by the same index.

2 Graph Transformations

In the first part of this chapter, the subset of graph transformations, modeling generation of an arbitrary *hp* refined mesh, based on initial mesh with horizontal sequence of triangular and rectangular finite elements, is presented. The described graph transformations can be generalized into a case of arbitrary two dimensional initial mesh. The process of the initial mesh generation is expressed by the graph transformations **P1**, **P2**, **P3** and **P29** presented in Fig. 1. The graph vertices with **I2B** and **IQ** labels represent rectangular finite elements, while the graph vertices with **IT1** and **IT2** labels represent triangular elements. Each triangular finite element consists in three vertices, three edge nodes and one interior node. Each rectangular finite element consists in four vertices, four edge nodes and one interior node. The graph grammar productions presented in Fig. 1 generates the topology of the mesh. This is followed by the generation of the structure of finite elements, which is expressed by graph grammar productions **P7** and **P8** presented in Figures 2 and 3 for triangular elements and by graph grammar production **P31** presented in Fig. 4 for rectangular elements.

Fig. 1. Graph grammar productions responsible for an initial mesh generation

Fig. 2. Productions generating the structure of a triangular element of the first type

Fig. 3. Productions generating the structure of a triangular element of the second type

The **V** label stands for an element vertex, **F** stands for an element edge (face), **I**, **I1** and **I2** stand for interiors for three types of elements, respectively. If an element is adjacent to mesh boundary, then its free bounds are connected to **B** labeled graph vertex, denoting the boundary, as well as to **F** labeled vertex, denoting missed second father of the edge (edges located inside the domain have two father elements). We can identify common edges of adjacent finite elements.

Fig. 4. Productions generating the structure of a rectangular initial mesh element

Fig. 5. Production for identification of the common edge of two adjacent triangular elements

The production **(P13)** that is identifying two adjacent triangular elements, and actually removing one duplicated edge, is presented in Fig. 5. The analogous production is utilize for identification of common edges for rectangular elements. The production **(P32)** that is identifying two adjacent elements, one rectangular and one triangular, and actually removing one duplicated edge, is presented in Fig. 6.

Once the structure of the mixed triangular and rectangular finite elements is generated, we can proceed with mesh refinements in the areas with strong singularities, where the numerical error is large. The decision about required refinements are made by the algorithm described in details in [8,9].

The refinement procedure is expressed by breaking element edges and interiors. To break a triangular element interior means to generate 4 new element interiors, and 3 new edges. To break a rectangular element interior means to

Fig. 6. Production for identification of the common edge of two adjacent elements, one rectangular and one triangular of the first type

Fig. 7. The *h* refinement of selected triangular and rectangular elements: breaking of element interiors followed by breaking of element edges

Fig. 8. Production breaking interior of first type triangular element

Fig. 9. Production breaking interior of second type triangular element

Fig. 10. Production for breaking rectangular element interior

generate 4 new element interiors, 4 new edges, and 1 new vertex. To break an element edge means to generate 2 new edge nodes and 1 new vertex. The procedure is illustrated in Fig. 7. The procedure of breaking of an element interior is expressed by (P14), (P15), (P34) productions presented in Figures 8-10. The newly created finite elements are represented by leaf nodes at the bottom level of generated refinement trees. The following mesh regularity rules are enforced during the process of mesh transformation, see [8]. An element edge can be broken only if two adjacent interiors have been already broken, or the edge

Fig. 11. Production allowing for breaking an element edge

Fig. 12. Production for breaking an element edge

is adjacent to the boundary. This is expressed by (P16) productions in Fig. 11. Analogous productions are utilized for mixed rectangular / triangular neighbors. The productions allows for breaking an element edge, by checking adjacent element interior. If two adjacent interiors are broken, the label of the element edge is changed from **F** to **E**. Only element edges denoted by **E** label can be broken. The physical breaking of an element edge is executed by graph grammar productions (P19) presented in Figure 12. An element interior can be broken only if all adjacent edges are of the same size as the interior. This is expressed by (P39) production in Fig. 13.

In other words, the history of refinements for adjacent edges is coded within the label of graph vertex representing element interior. The interior can be broken only after breaking adjacent edges and propagating the adjacency information along the refinement trees. The goal of the mesh regularity rule is to avoid multiple constrained edges, which leads to problems with approximation over such edges. The mesh regularity rule enforces breaking of large adjacent unbroken elements before breaking small element for the second time, which is illustrated

Fig. 13. Production allowing for breaking an element interior

Fig. 14. Enforcement of the one order irregularity mesh rule

Fig. 15. The initial mesh, the last mesh with 1% relative error of the solution, and the solution to the Fichera model problem

in Fig. 14. The mesh regularity rules are enforced on the level of graph grammar syntax.

3 Numerical Example

We conclude the presentation with the sequence of triangular finite element meshes generated for the L-shape domain model problem [8]. The problem consists in solving the Laplace equation $\Delta u = 0$ in Ω over the L-shape domain Ω presented on left panel in Fig. 15. The zero Dirichlet boundary condition $u = 0$ is assumed on the internal part of the boundary Γ_D . The Neumann boundary

condition $\frac{\partial u}{\partial n} = g$ is assumed on the external part of the boundary Γ_N . The temperature gradient in the direction normal to the boundary is defined in the radial system of coordinates with the origin located in the central point of the L-shape domain $g(r, \theta) = r^{\frac{2}{3}} \sin \frac{2}{3}(\theta + \frac{\pi}{2})$. The solution $u : R^2 \supset \Omega \ni u \mapsto R$ is a temperature distribution inside the L-shape domain. The initial mesh consists in 1 rectangular element and 4 triangular elements, see Fig. 17. The self-adaptive *hp*-FEM generates a sequence of meshes delivering exponential convergence of the numerical error with respect to the mesh size. The initial mesh, the last mesh with 1% relative error, and the solution, are presented in Fig. 17.

4 Conclusions

The CP-graph grammar is the tool for a formal description of mixed triangular and rectangular mesh transformations utilized by adaptive FEM. It models all aspects of the adaptive computations, including mesh generation, *h* and *p* refinements, as well as mesh regularity rules, including elimination of multiple constrained nodes. The technical nightmare with implementing the mesh regularity rules has been overcome by including the mesh regularity rules within the graph grammar syntax. The graph grammar have been formally validated by utilizing graph grammar definition software [5]. Our future work will concern several extensions of the presented CP-graph grammar. First, the graph grammar will be extended to model three dimensional meshes with tetrahedral, hexahedral, prism and pyramid elements, with an isotropic mesh refinements. Second, both the two and three dimensional graph grammars will be extended to support anisotropic mesh refinements. Moreover, the graph grammar will be extended to model the execution of the solver over the computational meshes, in three dimensions, for different kinds of elements.

Acknowledgments. The work of the second author has been supported by the Polish Ministry of Scientific Research and Information Technology and by the Foundation for Polish Science under Homming Programme.

References

1. Beal, M.W., Shephard, M.S.: A General Topology-Based Mesh Data Structure. *International Journal for Numerical Methods in Engineering* 40, 1573–1596 (1997)
2. Flasiński, M., Schaefer, R.: Quasi Context Sensitive Graph Grammars as a Formal Model of Finite Element Mesh Generation. *Computer Assisted Mechanics and Engineering Science* 3, 191–203 (1996)
3. Grabska, E.: Theoretical Concepts of Graphical Modeling. Part One: Realization of CP-Graphs. *Machine Graphics and Vision* 2(1), 3–38 (1993)
4. Grabska, E.: Theoretical Concepts of Graphical Modeling. Part Two: CP-Graph Grammars and Languages. *Machine Graphics and Vision* 2(2), 149–178 (1993)
5. Grabska, E., Hliniak, G.: Structural Aspects of CP-Graph Languages. *Schedae Informaticae* 5, 81–100 (1993)

6. Paszyński, M., Paszyńska, A.: Graph transformations for modeling parallel *hp*-adaptive Finite Element Method. In: Wyrzykowski, R., Dongarra, J., Karczewski, K., Wasniewski, J. (eds.) PPAM 2007. LNCS, vol. 4967, pp. 1313–1322. Springer, Heidelberg (2008)
7. Paszyńska, A., Paszyński, M., Grabska, E.: Graph transformations for modeling *hp*-adaptive Finite Element Method with tringular elements. In: Bubak, M., van Albada, G.D., Dongarra, J., Sloot, P.M.A. (eds.) ICCS 2008, Part III. LNCS, vol. 5103, pp. 604–613. Springer, Heidelberg (2008)
8. Demkowicz, L.: Computing with *hp*-Adaptive Finite Elements. Chapman & Hall/Crc Applied Mathematics & Nonlinear Science, vol. I (2006)
9. Paszyński, M., Kurtz, J., Demkowicz, L.: Parallel Fully Automatic *hp*-Adaptive 2D Finite Element Package. Computer Methods in Applied Mechanics and Engineering 195(7-8)(25), 711–741 (2006)