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

Anna Paszyńska¹, Maciej Paszyński², and Ewa Grabska¹

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

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

Abstract. The paper presents composition graph (CP-graph) grammar, which consists of a set of CP-graph transformations, suitable for modeling triangular finite element mesh transformations utilized by the self-adaptive *hp* Finite Element Method (FEM). The *hp* adaptive FEM allows to utilize distributed computational meshes, with finite elements of various size (thus *h* stands for element diameter) and polynomial orders of approximation varying locally, on finite elements edges and interiors (thus *p* stands for polynomial order of approximation). The computational triangular mesh is represented by attributed CP-graph. The proposed graph transformations model the initial mesh generation, procedure of *h* refinement (breaking selected finite elements into son elements), and *p* refinement (adjusting polynomial orders of approximation on selected element edges and interiors). The graph grammar has been defined and verified by implemented graph grammar transformation software tool.

1 Introduction

The Composite Programable graph grammar (CP-graph grammar) was proposed in [1], [2], [3] as a tool for formal description of various design processes. The CP-graph grammar expresses a design process by graph transformations executed over the CP-graph representation of the designed object. In this paper, a new application of the CP-graph grammar is proposed, where the grammar is utilized to describe mesh transformations occurring during the self-adaptive *hp* Finite Element Method (FEM) calculations. The paper is an extension of the CP-graph grammar model introduced in [4], [5] for rectangular finite element meshes. The developed self-adaptive *hp* FEM [6], [7], [8], [9], [10] is an evolvable system that generates a sequence of optimal *hp* meshes, with finite elements of various size and polynomial orders of approximations varying locally on finite element edges, faces and interiors. The generated sequence of meshes delivers

exponential convergence of the numerical error with respect to the mesh size (number of degrees of freedom or CPU time). Each *hp* mesh is obtained by performing a sequence of *h* and *p* refinements on the predefined initial mesh. The *h* refinement consists in breaking selected triangular finite element into four son elements. The *p* refinement consists in increasing polynomial order of approximation on selected edges, faces and/or interiors. The process of generation of an

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

hp mesh is formalized by utilizing CP-graph grammar. This allows to express mesh transformation algorithms by means of graph transformations managed by control diagrams. This involves the generation of the initial mesh, as well as *h* and *p* refinements. The formalization of the process of mesh transformation allows to code the mesh regularity rules on the level of graph grammar syntax, which simplifies the algorithms and prevents computational mesh to be inconsistent.

The computational triangular mesh is represented by attributed CP-graph. 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 (bounds connected to external vertices). The corresponding free in/out bounds have the same number on both sides of a production. Thus, the embedding transformation is coded in the production by assuming the same number of free bounds on both sides of the production. The execution of graph transformations are controlled by control diagrams prescribing the required order of productions. The presented graph grammar has been defined and verified by the implemented software tool.

2 Graph Transformations for Modeling the Initial Mesh Generation

In this section, the subset of graph transformations, modeling generation of an arbitrary hp refined mesh, based on initial mesh with horizontal sequence of triangular 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 presented in Fig. 1. There are two types of triangular finite elements, presented in Fig. 2. Each triangular finite element consists in three vertices, three edge nodes and one interior node.

Fig. 2. Two types of triangular finite elements. Each element consists in 3 vertices, 3 edge nodes and one interior node.

Once the sequence of initial mesh elements is generated, the structure of each element is produced, as it is expressed by productions presented in Fig. 3. The V label stands for an element vertex, F stands for an element edge (face), $I1$ and $I2$ stand for interiors for two 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 FAL labeled vertex, denoting missed second father of the edge (edges located inside the domain have two father elements). There is the similar production for the second element type. There are also similar productions for elements $E1B$, $E2B$, $E1E$, $E2E$ located at the beginning and the end of the sequence.

We can identify common edges of adjacent finite elements. The production that is identifying two adjacent elements, and actually removing one duplicated edge, is presented in Fig. 4. The identification is possible, since we keep finite elements connectivity at the top level of the graph.

3 Graph Transformations for Modeling Mesh Refinements

Once the structure of triangular 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 is made by knowledge driven artificial intelligence algorithm [6], [7], [8], [9], [10]. The selected finite elements can be either h , p or hp refined.

Fig. 3. Production for generating the structure of the first element type

Fig. 4. Production for identifying the common edge of two adjacent elements

The *h* refinement is expressed by breaking element edges and interiors. To break an element interior means to generate four new element interiors, and three new edges, as it is illustrated on middle panel in Fig. 5. To break an element edge

Fig. 5. The h refinement of a triangular element: breaking of the element interior followed by breaking of the element edges

Fig. 6. Graph grammar production for breaking element interior, for the element of the first type

means to generate two new edge nodes and one new vertex, as it is illustrated on right panel in Fig. 6 for all three edges of the original triangular element. These procedures are expressed by (PI12-15) productions presented in Figs. 3-4. The newly created finite elements are never stored in the data structure. They are dynamically localized at the bottom level of generated refinement trees.

The following mesh regularity rules are enforced during the process of mesh transformation, see [6]. An element edge can be broken only if two adjacent interiors have been already broken, or the edge is adjacent to the boundary. This is expressed by (PI14) and (PI15) productions in Figs. 8 and 9. An element interior can be broken only if all adjacent edges are of the same size as the interior. This is expressed by (PI16) production in Fig. 10. The graph vertex JI4 is obtained after breaking all adjacent edges, and propagating adjacency information along the refinement tree, from the father node down to children nodes. The propagation of the adjacency data is obtained by executing six productions, related

to three different edges of two types of elements. One of these productions is presented in Fig. 11. In other words, the history of adjacent edges refinements 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

Fig. 7. Graph grammar production for breaking an element interior, for the element of the second type

avoid multiple constrained edges, which leads to problems with approximation over such an edge. The mesh regularity rule enforces breaking of large adjacent unbroken elements before breaking small element for the second time, which is illustrated in Fig. 12. The mesh regularity rules are enforced on the level of graph grammar syntax.

4 Numerical Example

We conclude the presentation with the sequence of triangular finite element meshes generated for the L-shape domain model problem [6]. The problem consists in solving the Laplace equation

$$\Delta u = 0 \text{ in } \Omega \tag{1}$$

over the L-shape domain Ω presented in Fig. 13. The zero Dirichlet boundary condition

$$u = 0 \text{ on } \Gamma_D \tag{2}$$

is assumed on the internal part of the boundary Γ_D . The Neumann boundary condition

$$\frac{\partial u}{\partial n} = g \text{ on } \Gamma_N \tag{3}$$

Fig. 8. Graph grammar production allowing for breaking an element edge, for the edge surrounded by two broken interiors

Fig. 9. Graph grammar production for actual breaking of an element edge

Fig. 10. Graph grammar production allowing for breaking an element interior

Fig. 11. One of productions executing propagation of the adjacency data along the refinement tree

Fig. 12. Mesh regularity rule enforces breaking of large adjacent element before breaking of small element

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.

$$g(r, \theta) = r^{\frac{2}{3}} \sin \frac{2}{3} \left(\theta + \frac{\pi}{2} \right). \quad (4)$$

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 six triangular finite elements, presented on the first panel in Fig. 13. The self-adaptive *hp*-FEM code generates a sequence of meshes delivering exponential convergence of the numerical error with respect to

Fig. 13. The sequence of triangular finite element meshes for the L-shape domain problem

the problem size. The initial mesh, the second mesh, the optimal mesh delivering less than 5% relative error accuracy of the solution, and the solution over the optimal mesh are presented in Fig. 13.

5 Conclusions

The CP-graph grammar is the tool for a formal description of triangular 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 [3]. The graph grammar can be easily extend to support anisotropic mesh refinements and three dimensional computations.

Acknowledgments. The work reported in this paper was supported by Polish MNiSW grant no. 3 TO 8B 055 29. The work of the first author has been also supported by the Foundation for Polish Science under Homming Programme.

References

1. Grabska, E.: Theoretical Concepts of Graphical Modeling. Part One: Realization of CP-Graphs. *Machine Graphics and Vision* 2(1), 3–38 (1993)
2. Grabska, E.: Theoretical Concepts of Graphical Modeling. Part Two: CP-Graph Grammars and Languages. *Machine Graphics and Vision* 2(2), 149–178 (1993)
3. Grabska, E., Hliniak, G.: Structural Aspects of CP-Graph Languages. *Schedae Informaticae* 5, 81–100 (1993)
4. Paszyński, M., Paszyńska, A.: Graph transformations for modeling parallel hp -adaptive Finite Element Method. In: *Parallel Processing and Applied Mathematics*, Gdańsk, Poland, September 2007. LNCS, vol. 4967, Springer, Heidelberg (in press, 2008)
5. Paszyński, M.: Parallelization Strategy for Self-Adaptive PDE Solvers. *Fundamenta Informaticae* (submitted, 2007)

6. Rachowicz, W., Pardo, D., Demkowicz, L.: Fully Automatic *hp*-Adaptivity in Three Dimensions. ICES Report 04-22, pp.1–52 (2004)
7. Demkowicz, L.: Computing with *hp*-Adaptive Finite Elements, vol. I. Chapman & Hall/Crc Applied Mathematics & Nonlinear Science, New York (2006)
8. Demkowicz, L., Pardo, D., Paszyński, M., Rachowicz, W., Zduneka, A.: Computing with *hp*-Adaptive Finite Elements, vol. II. Chapman & Hall/Crc Applied Mathematics & Nonlinear Science, New York (2007)
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), 711–741 (2007)
10. Paszyński, M., Demkowicz, L.: Parallel Fully Automatic *hp*-Adaptive 3D Finite Element Package. Engineering with Computers 22(3-4), 255–276 (2006)