Association for Information Systems
AIS Electronic Library (AISeL)

SAIS 2018 Proceedings | Southern (SAIS)

Spring 3-23-2018

A Graphical User Interface for Designing Graph Grammars

Patrick Andrade
Vibra-Tech Inc., patandrade.55@gmail.com

Yingfeng Wang
Middle Georgia State University, yingfeng.wang@mga.edu

A GRAPHICAL USER INTERFACE FOR DESIGNING GRAPH GRAMMARS

Patrick Andrade
Vibra-Tech Inc.
patandrade.55@gmail.com

Yingfeng Wang
School of Information Technology
Middle Georgia State University
yingfeng.wang@mga.edu

ABSTRACT

Graph grammar has been widely applied in many scientific areas. However, designing graph grammar is very challenging for users without strong computer science background. This paper presents a graphical user interface (GUI) for designing graph grammars following an edge-based context-sensitive graph grammar formalism, EGG. This GUI significantly eases graph grammar design, especially for users unfamiliar with the grammar format.

Keywords

Graph Grammar, GUI

INTRODUCTION

Many scientific areas represent problems by using simple graphs composed of vertices and edges. These graphs can be effectively processed by graph grammar, a powerful computational tool. Graph grammar can be considered as an extension of formal grammar. Like a formal grammar, a graph grammar is composed of one or more production rules. A formal grammar can process strings using derivation and parsing operations. In contrast, a graph grammar can process simple graphs using these two types of operations. All graph grammar formalisms fall into two categories: context-free and context-sensitive. The former only allows a non-terminal on the left side of each production rule and is relatively simple (Drewes et al., 1997; Janssens, 1982; Rozenberg and Welzl, 1986; Wittenburg, 1992), while the latter allows a graph on the left side and has stronger expressive power (Golin, 1991; Kong et al., 2006; Marriott, 1994; Rekers and Schürr, 1997). This paper focuses on an edge-based context-sensitive graph grammar formalism, EGG (Zeng et al., 2008).

This work aims at providing a convenient graph design tool for EGG. EGG can process simple graphs that are composed of vertices and edges. In a simple graph, each edge is associated with two vertices. The EGG formalism accepts edges with one or two specified vertices. The former are dangling edges, while the latter are regular edges. These two edge types provide necessary flexibility for grammar design. EGG has been used in some scientific areas besides computer science, e.g., analytical chemistry (Wang et al., 2017). Therefore, we expect many users are unfamiliar with graph grammar. However, grammar design is very challenging for users without the expertise of graph grammar. An EGG based tool usually requires grammars to be stored in a text file with a required format, e.g., the standard GML format (Himsolt, 1999). There are many technical details involved in this text file. These details are very confusing to users who are unfamiliar with graph grammar. This is a bottleneck of applying graph grammar to solve real problems. Here, we propose to use a graphical editor for designing graph grammars, because a graphical editor can hide many technical details. There are some powerful graphical editors such as yEd and Microsoft Visio. These editors provide rich features for designing graphs but cannot covert designed graphs into the file format used for graph grammar. Therefore, it is necessary to develop a specific graphical editor for addressing this issue. This paper presents GEGG, a graphical user interface (GUI), for designing EGG grammars. The major feature of GEGG is to allow users to draw EGG grammars like drawing graphs and export designed grammars in the standard GML format, which is a popular format of computational tools using EGG. GEGG removes the hurdle of using EGG. It will facilitate the development and application of graph grammar.

METHODS

In order to lower the difficulty of grammar design, we developed GEGG as a GUI of designing EGG grammars. GEGG is implemented in Java. It is composed of two parts. One is the graphical user interface for users to design grammars like drawing graphs. The other is the module for automatically converting graphs into GML format. The interface allows users to

use the mouse to create vertices and edges. Users can click the corresponding button for specifying which type of graphical elements should be created in the next action. Existing vertices and edges can also be edited. For example, users can add comments to related vertices and edges. When drawing an edge, users also need to draw two associated vertices. Both vertices of a regular edge should be specified, so users need to use solid cycles for both vertices. However, one vertex of a dangling edge is unspecified. To highlight the existence of this vertex, users are suggested to use a dashed circle for drawing this vertex. Therefore, we can distinguish between specified and unspecified vertices according to circle types.

To meet different needs of output files, GEGG supports two file formats. One is an internal format. It is only used by GEGG for saving intermediate designs. Users can save their current works in files with this format. GEGG can resume these works in the future by loading the saved files. The other format is the standard GML format (Himsolt, 1999). After finishing grammar design, users can export the designs as GML files. Applications using EGG grammars, e.g., MIDAS-G (Wang et al., 2017), which is a metabolomics tool, can directly use these GML files. It is worth noting that the GML format loses some design information, e.g., the positions of edges and vertices. So, GEGG cannot load GML files.

Here, we use an example to demonstrate how to use GEGG. Figure 1 is a screenshot of designing a grammar rule in GEGG. Solid circles refer to specified vertices, while dash circles refer to unspecified vertices. Lines refer to edges. Users can add and edit comments of all edges and vertices. Besides editing comments, users can finish grammar design by using mouse.

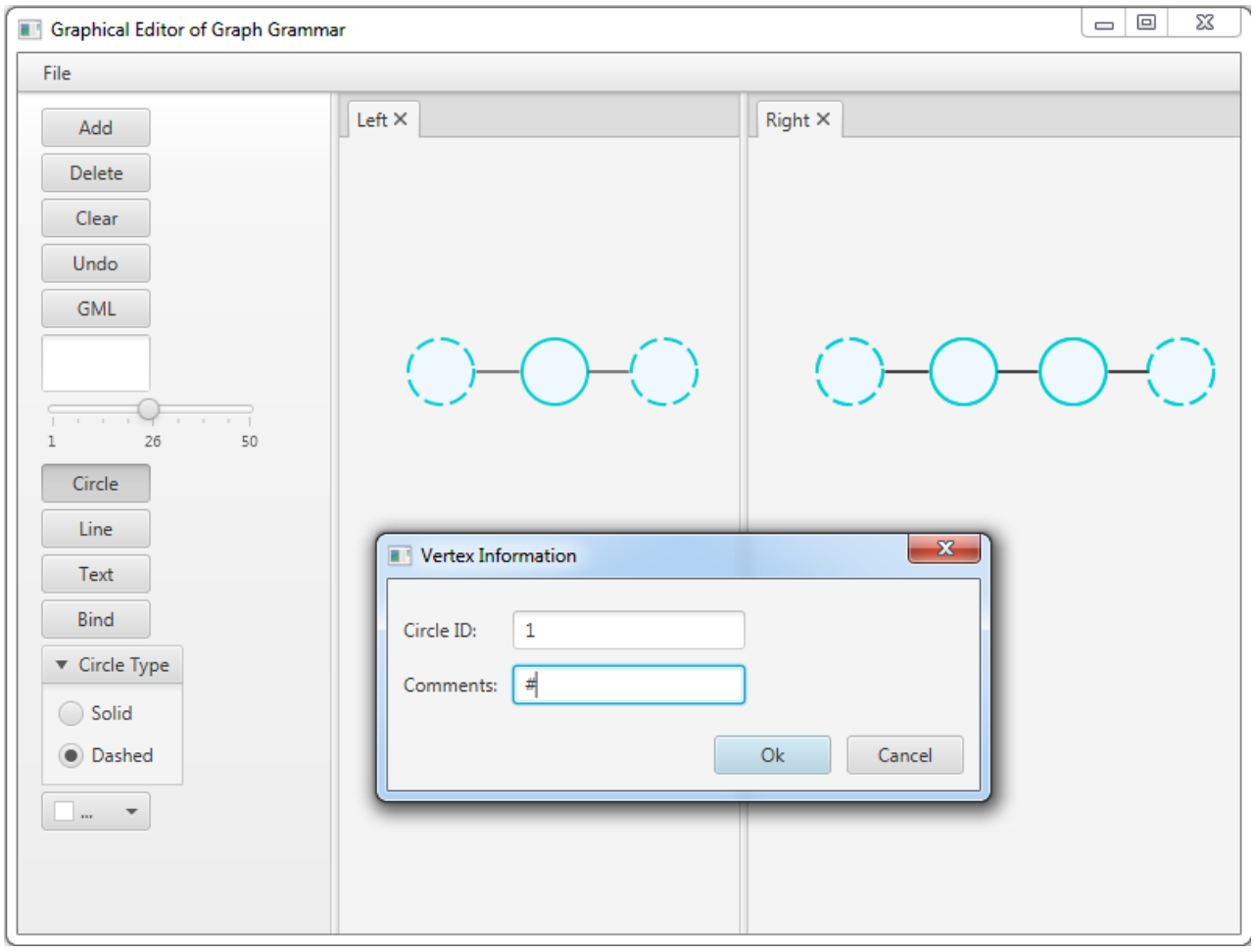

Figure 1. The screenshot of GEGG Interface. This is an example of editing a grammar rule. Solid circles refer to specified vertices, while dash circles refer to unspecified vertices. Lines refer to edges. Users can add and edit comments of all edges and vertices. For example, the current popup window is for the comments of the middle cycle on the left side.

CONCLUSION

In this paper, we present a GUI for designing graph grammars. This tool can significantly ease grammar design. The future work will add more features to address specific needs. For example, EGG has been applied in metabolomics (Wang et al.,

2017). We will also integrate GEGG into the corresponding computational tool by implementing features related to metabolomics.

REFERENCES

1. Drewes, F., Habel, A., and Kreowski, H. J. (1997) Hyperedge replacement graph grammars. *Handbook of Graph Grammars and Computing by Graph Transformation*, 1, 95–162.
2. Golin, E. J. (1991) A method for the specification and parsing of visual languages, Brown University.
3. Himsolt, M. (1999) GML : A portable graph file format. *Technical Report*. Passau, Germany.
4. Janssens, D. (1982) Graph grammars with neighbourhood-controlled embedding, *Theoretical Computer Science*, 21, 1, 55–74.
5. Kong, J., Zhang, K., and Zeng, X. (2006) Spatial graph grammars for graphical user interfaces, *ACM Transactions on Computer-Human Interaction*, 13, 2, 268–307.
6. Marriott, K. (1994) Constraint multiset grammars. *In IEEE Symposium on Visual Languages*, 118–125.
7. Rekers, J., and Schürr, A. (1997) Defining and parsing visual languages with layered graph grammars, *Journal of Visual Languages and Computing*, 8, 1, 27–55.
8. Rozenberg, G., and Welzl, E. (1986) Boundary NLC graph grammars–basic definitions, normal forms, and complexity. *Information and Control*, 69, 1-3, 136–167.
9. Wang, Y., Wang, X., and Zeng, X. (2017) MIDAS-G: A computational platform for investigating fragmentation rules of tandem mass spectrometry in metabolomics, *Metabolomics*, 13, 10, 116.
10. Wittenburg, K. (1992) Earley-style parsing for relational grammars, *In IEEE Workshop on Visual Languages*, 192–199.
11. Zeng, X.-Q., Han, X.-Q., and Zou, Y. (2008) An Edge-Based Context-Sensitive Graph Grammar Formalism, *Journal of Software*, 19, 8, 1893–1901.