

# A Demonstration of an Efficient Tool for Graph Matching and Transformation

Khalil Drira <sup>\*,\*\*</sup>, Ismael Bouassida Rodriguez <sup>\*,\*\*</sup>

<sup>\*</sup>CNRS ; LAAS ; 7 Avenue du Colonel Roche, F-31077 Toulouse, France

<sup>\*\*</sup>Université de Toulouse ; UPS, INSA, INP, ISAE ; LAAS ; F-31077 Toulouse, France  
khalil@laas.fr

<http://homepages.laas.fr/khalil/page>

**Résumé.** Nous avons implanté un outil efficace de recherche de morphismes et de transformation de grands graphes et l'avons utilisé pour modéliser les architectures dynamiques, les activités coopératives et d'autres applications. L'outil est en cours d'extension avec des interfaces graphiques et un module d'interopérabilité en XML. Une version préliminaire est disponible sous <http://homepages.laas.fr/khalil/GTE/>

## 1 Description

GTE, the graph matching and transformation engine is an efficient tool we have been implementing in C++ since a decade now. It is an efficient implementation of an extension of Messmer's algorithm (see Messmer (1995)). Our experiments presented in Bouassida-Rodriguez et al. (2008) show that the tool is capable of searching small and medium graph patterns in huge graphs in a short time. A computational complexity analysis of our algorithm has conducted and performant experimental results are obtained. We have also shown that, when only constant labels are considered, this complexity is similar to the complexity of Ullmann's algorithm (see Ullmann (1976)). Both pattern graph (called rule graph) and host graph have labelled nodes and edges. The rule graph labels may be totally or partially instantiated. Unification is conducted for non-instantiated labels.

The tool can be used non-interactively as a C++ library providing a function that can be invoked from either a C++ or a Java main program. The tool can be used through as a C++ executable that reads rule graph and host graph description from input TXT or XML files. It has been recently associated with a graphical user interface (Figure 1) composed of the following zones and components :

- A menu bar offering to the user many items to manipulate the interface contents such as creating, deleting, saving projects, graphs and rules.
- A tools bar that the user can use to edit graphs and rules (saving, undo, redo...).
- Project Explorer giving the user a tree representing the list of opened projects, graphs and rules.
- A components panel containing a list of buttons for creating nodes and edges.
- A graph representing zone offering to the user the possibility to open and show graphs she/he is manipulating.

## A Demonstration of an Efficient Tool for Graph Matching and Transformation

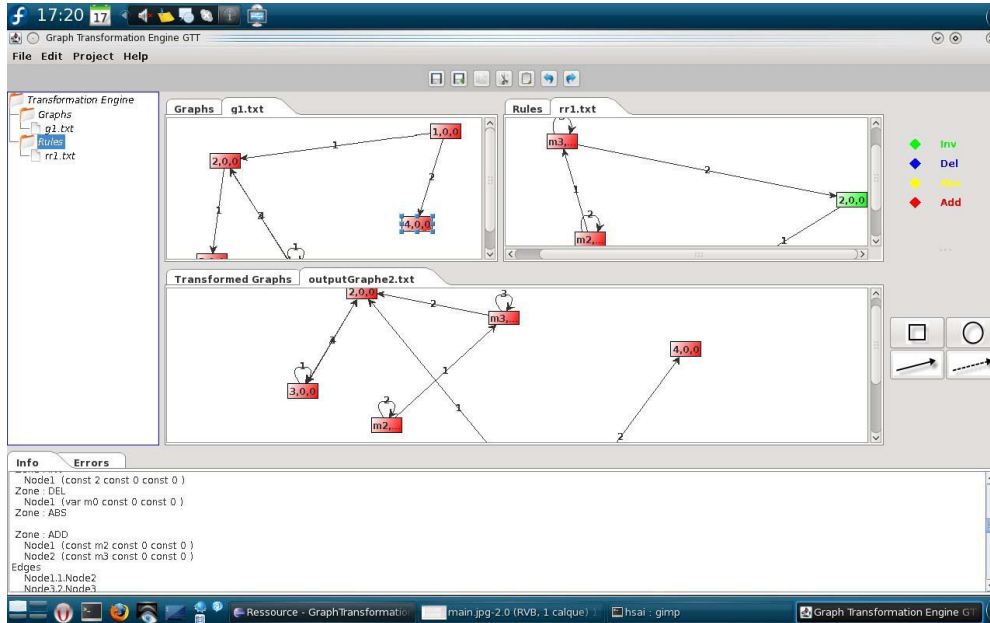


FIG. 1 – *The Graphical User Interface*

- A rule representing zone offering to the user the possibility to open and show rules she/he is manipulating.
- A transformed graph zones offering to the user the possibility to open and show graphs she/he had transformed.
- A rule legend with which the user can distinguish between rule zones (*Inv*, *Del*, *Abs*, *Add*).
- Two tabs showing to the user information and errors when transforming a graph.

The user can export graphs and rules from the application to TXT or XML according to the standard RuleML format as well as an image. The interface offers an export wizard which gives the user the possibility to specify file name, directory where to export and the export format. The exported XML graph file an example of which is shown in Figure 2 is composed of a Graph element containing a list of nodes and edges Elements with different attributes describing each element. The exported XML rule file is composed of a Rule element containing a list of nodes representing the different zones of the rule (*Inv*, *Del*, *Abs*, *Add*). Each zone element is composed of a graph containing a list of node and edge Elements with different attributes describing each one.

## 2 Acknowledgment

Have contributed to this work : Ismael Bouassida Rodriguez, Khalil Drira, Rony Ghostine, Karim Guennoun, Moncef Sadoq and Hachemi Sai.

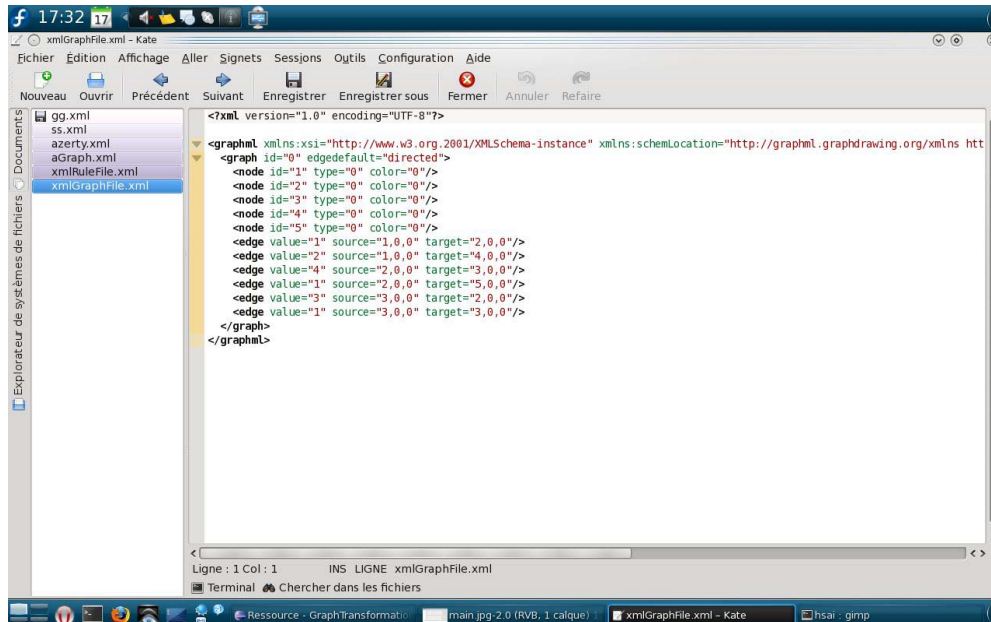


FIG. 2 – The XML window

## Références

- Bouassida-Rodriguez, I., K. Guennoun, K. Drira, C. Chassot, et M. Jmaiel (2008). Implementing a rule-driven approach for architectural self configuration in collaborative activities using a graph rewriting formalism. In *CSTST '08 : Proceedings of the 5th international conference on Soft computing as transdisciplinary science and technology*, New York, NY, USA, pp. 484–491. ACM.
- Messmer, B. (1995). *Efficient Graph Matching Algorithms for Preprocessed Model Graphs*. Ph. D. thesis, Institut für Informatik und angewandte Mathematik, Universität Bern, Switzerland.
- Ullmann, J. R. (1976). An algorithm for subgraph isomorphism. *Journal of the ACM* 23(1), 31–42.

## Summary

We implemented an efficient tool for graph matching and transformation. We have used it to model dynamic architectures, co-operative activities and other applications. The tool is being extended with graphical user interfaces and XML interoperability. A preliminary version is available under <http://homepages.laas.fr/khalil/GTE/>