Parallel Graph Transformation Applied to AGG Tool

Asmaa Aouat
Department of computer engineering
University of Science and Technology- Mohamed Boudiaf
Oran, Algeria
asmaaoran@hotmail.fr

El Abbassia Deba
Department of computer engineering
University of Oran
Oran, Algeria
abdeba@gmail.com

Abstract— Graph transformation is one of the key concepts in graph grammar. In order to accelerate the graph transformation, the concept of parallel graph transformation has been proposed by different tools such as AGG tool. The theory of parallel graph transformation used by AGG just allows clarifying the concepts of conflict and dependency between the transformation rules. This work proposes an approach of parallel graph transformations which enables dependent transformation rules to be executed in parallel.

Keywords-component; Graph transformation; Critical pair; Synchronization; AGG.

I. Introduction (Heading 1)

In the late '60s, graph transformation was motivated by considerations about pattern recognition, compiler construction, and data type specification. Since then the list of areas which have interacted with the development of graph transformation has grown impressively. Besides the areas mentioned, it includes the software specification and software development, database design, model transformation, computer animation, biology development, music composition, visual languages, and many others [1].

The wide applicability of graph transformation is due to the fact that graphs are a natural way of describing complex situations on an intuitive level. Basically, three approaches can be distinguished in the graph transformation, which are *Node Replacement Graph Grammars, Hyperedge Replacement Graph Grammars* and *the algebraic approach.* Generally, the first two approaches are used in the fields of biology and chemistry in contrast to the algebraic approach that is widely used in the Model Driven Engineering (MDE) [2, 3].

When graph transformation is used to describe concurrent complex systems where graph productions are independent, the used techniques of graph transformation are not always sufficient. There exist different techniques to accelerate the complex graph transformation, such as parallel graph transformation [4]. Therefore our paper, propose an efficient technique of parallel graph transformation under AGG tool, which supports the “cases of dependence and independence between the transformation rules” for avoiding the blocking created by AGG in the case of conflict between the dependent rules.

The paper is organized as follows: After the introduction, section 2 reviews the basic concepts for graph transformation. The third section describes the parallelism in the AGG tool through a case study to showing its limits. In the fourth section, we propose like generalized solution the approach of synchronized graph transformation. The fifth section discusses related work. In the end, a conclusion finishes this paper and presents future directions.

II. Graph transformation:General concepts

As we have already mentioned, there are different approaches of graph transformation, but we retain the algebraic approach in this work, because it is interesting from the theoretical view point and is considered as a formal transformation approach based on attributed graph grammars where validation of the transformation is verified.

In analogy with Chomsky grammars, graph grammars are used to describe graph transformations or to generate valid sets of graphic productions. A graph grammar of the algebraic approach is made of an initial graph and a set of transformation rules. Every transformation rule incorporates a left hand side rule (LHS) and a right hand side rule (RHS). When a match is found between the LHS of a rule and a part of the initial graph, the subgraph is replaced by the corresponding part of the RHS rule. The rules may also have a condition that must be satisfied in order to apply the rule. The application of a rule is called a derivation that allows the passage from a graph to another. Figure 1 illustrates the principle of graph transformation [1].

III. Parallel transformation by AGG

In this section, we study under what conditions two graph transformation rules can be applied in parallel. This leads to the concepts of parallel and sequential independence of graph transformation rules. The definition of these conditions is presented in the Local Church-Rosser theorem. Before discussing Local Church-Rosser theorem, we must firstly define parallel and sequential independence of two graph transformation rules [3].

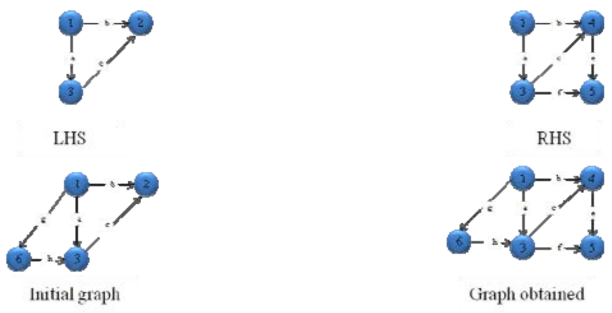


Figure 1. Basic concepts of graph transformation

A. *Conflict and dependance between transformation rules*

Parallel independence Two graph transformation rules G ⇒H_1 and G ⇒H_2 are parallel independent if there exist two morphisms i : L_1 →D_2 and j : L_2 →D_1 such that $f_2 \circ i = m_1$ and $f_1 \circ j = m_2$.

$$\begin{tikzcd}
R_1 & K_1 \arrow[l, "r_1"'] \arrow[r, "l_1"] \arrow[d, "k_1"] & L_1 \arrow[dr, "m_1"] \arrow[drrr, dashed, "i"] & & L_2 \arrow[dl, "m_2"'] \arrow[dlll, dashed, "j"'] & K_2 \arrow[l, "l_2"'] \arrow[r, "r_2"] \arrow[d, "k_2"] & R_2 \arrow[d, "n_2"] \\
H_1 \arrow[from=u, "n_1"] & D_1 \arrow[l, "g_1"] \arrow[rr, "f_1"] & & G & & D_2 \arrow[ll, "f_2"'] \arrow[r, "g_2"] & H_2
\end{tikzcd}$$

Sequential independence Two graph transformation rules G ⇒H⇒G' are sequentially independent if there exist two morphisms i : R_1 →D_2 and j: L_2 →D_1 such that $f_2 \circ i = n_1$ and $g_1 \circ j = m_2$.

$$\begin{tikzcd}
L_1 & K_1 \arrow[l, "l_1"'] \arrow[r, "r_1"] \arrow[d, "k_1"] & R_1 \arrow[dr, "n_1"] \arrow[drrr, dashed, "i"] & & L_2 \arrow[dl, "m_2"'] \arrow[dlll, dashed, "j"'] & K_2 \arrow[l, "l_2"'] \arrow[r, "r_2"] \arrow[d, "k_2"] & R_2 \arrow[d, "n_2"] \\
G \arrow[from=u, "m_1"] & D_1 \arrow[l, "f_1"] \arrow[rr, "g_1"] & & H & & D_2 \arrow[ll, "f_2"'] \arrow[r, "g_2"] & G'
\end{tikzcd}$$

Intuitively, two independent graph transformations are parallel if their correspondence (match) does not overlap on the elements that are preserved by the second transformation. An intuitive case that can be given, if neither of the two transformations doesn't remove an item preserved by another transformation.

Local Church-Rosser theorem

Given two parallel independent graph transformations G ⇒H_1 and G ⇒H_2, there exist a graph G' and two graph transformations H_1⇒ G' and H_2⇒ G' such that G ⇒H_1⇒G' and G⇒H_2⇒G' are sequentially independent.

Given two sequentially independent graph transformations G ⇒H_1 ⇒G’, there exist a graph H_2 and a graph transformations G ⇒H_2 ⇒G’ such that G ⇒H_1 and G ⇒H_2 are parallel independent [3].

$$\begin{tikzcd}
 & G \arrow[dl, Rightarrow, "{p_1, m_1}"'] \arrow[dr, Rightarrow, "{p_2, m_2}"] & \\
H_1 \arrow[dr, Rightarrow, "{p_2, m_2'}"'] & & H_2 \arrow[dl, Rightarrow, "{p_1, m_1'}"] \\
 & G' &
\end{tikzcd}$$

Example of dependence between two transformation rules

Figure 2 shows an example of two transformation rules P_1 and P_2 that are both parallel and sequentially independent. So the left rule P_1 and the right rule P_2 satisfy the condition for they can be applied on the same graph.

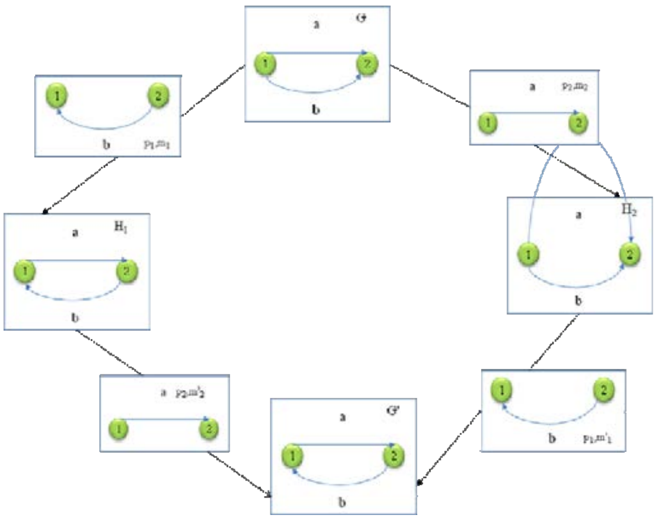


Figure 2. Independence between transformation rules

B. Case study

In order to explain different aspects of parallel graph transformation in the case of dependence between transformation rules and to illustrate our approach, we introduce a simple example of a business center. Its graphical diagram is shown in Figure 3 like an instance of class diagram.

The ***Shop*** offers a ***Cart*** of shopping for ***Clients*** to transport the ***Wares***. The Clients carry a certain amount of cash; will be cashed at the ***Cash desk***.

Bill lists the wares collected by the Client together with the overall amount of the prizes. Since we are about to use a class diagram, specifying only class, associations, attributes, and constraints [5]. An instance of this class diagram represents our graph.

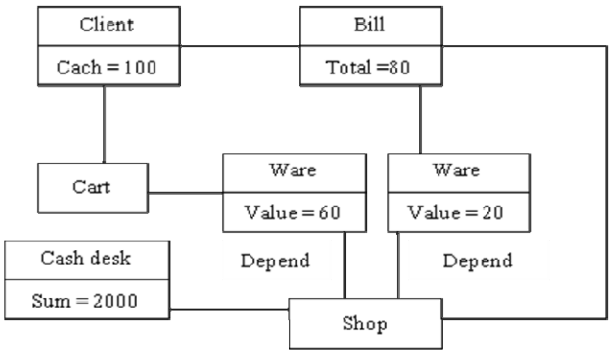


Figure 3. Instance of class diagram for the business center

For shopping in the Shop, the Clients take a Cart; make the choice of Wares by selecting to ***Rack*** of Shop and placing them in the Cart. Once the selection of Wares is completed, the Clients proceed toward the Cash desk. There, he finds a clerk is waiting to sell the Wares. The Client withdraws the Wares progressively from the Cart and present then to clerk which establishes Bill for the list of Wares. The total of the Bill is increased by the Wares prizes, added. The ownership of the Wares is transferred from the Shop to the Client, as described by the ***Depend*** links from the Shop. These facts are illustrated in Figure 4 in the form of a set of transformation rules.

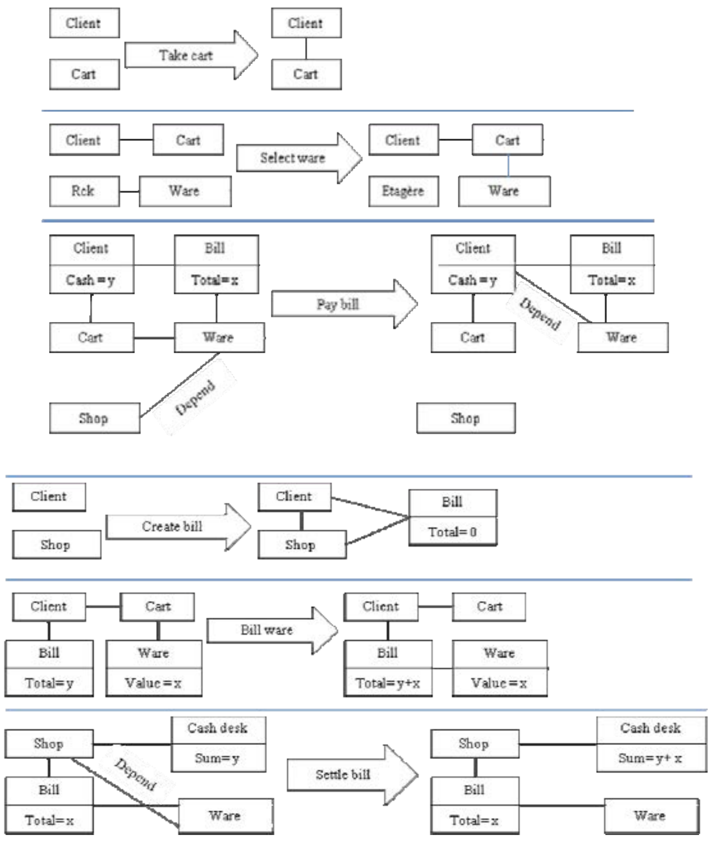


Figure 4. Set of transformation rules for business center

An example of a conflict between two transformation rules of ***pay bill*** and ***settle bill*** is given in Figure 5. The two transformation rules share and delete ***Depend*** link between ***Ware*** and ***Shop***. Thus, they overlap in items that are deleted. As a consequence, each of the two disables the other one, i.e., there is dependence between these two rules, and then they cannot be parallelized, unlike the independence between P_1 and P_2 shown in Figure 2.

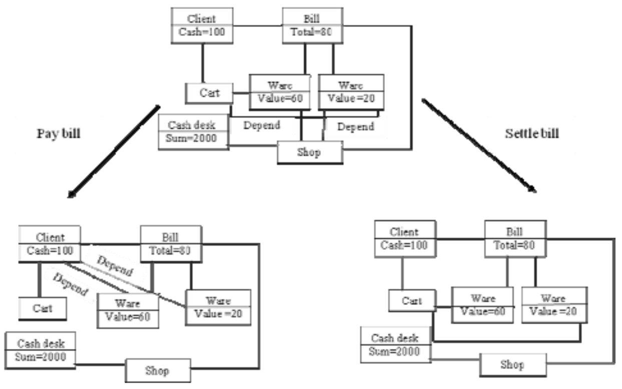
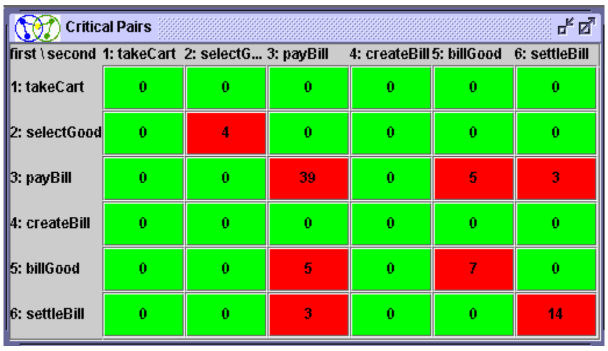


Figure 5. The conflict between pay bill and settle bill

AGG (Attributed Graph Grammar) is a tool well appropriate for graph transformation systems supporting the algebraic approach. It was developed and expanded over the past 15 years, and implements the algebraic approach of "Single Pushout Approach" SPO like transformation behavior. Currently, AGG supports the computation of critical pairs for attributed graphs. All transformation rules that overlap or trigger a conflict i.e.; which are dependent between them, are detected by AGG via the critical pairs. The critical pair analysis is offered through a graphical user interface to browse through the computed pairs [6]. Figure 6 shows a screen dump of all critical pairs that are analyzed by AGG for the two transformation rules ***pay bill*** and ***settle bill*** where the essence of this critical pairs is the Depend link between the Shop and the Ware.

Critical pairs seem a good way to analyze and detect conflicts between transformation rules but in reality they represent an obstacle for independent transformation rules because they interrupt the graph transformation. As a conclusion, AGG doesn't allow parallel execution of dependent transformation rules. Hence our interest appears at this level to resolve this issue in the next section.


Critical Pairs

first \ second	1: takeCart	2: selectG...	3: payBill	4: createBill	5: billGood	6: settleBill
1: takeCart	0	0	0	0	0	0
2: selectGood	0	4	0	0	0	0
3: payBill	0	0	39	0	5	3
4: createBill	0	0	0	0	0	0
5: billGood	0	0	5	0	7	0
6: settleBill	0	0	3	0	0	14

Figure 6. Critical pair

IV. SYNCHRONIZED GRAPH TRANSFORMATION

The objective of our approach is to propose a solution for blocking provided by critical pairs in the case of conflict in AGG tool. Our approach uses graph transformation that is both synchronous and asynchronous and meets the needs of parallel applications under AGG tool. Synchronous and asynchronous actions are usually distinguished according to their interaction type. Synchronized actions are executed when the transformation rules are dependent between then while asynchronous actions also enable transformation to independent transformation rules [7].

In reality, the rules are not independent of each other can still be applied in a parallel way, if they can be synchronized by sub- rule. If two actions include the deletion or creation of the same node or same edge, this operation can be encapsulated in a separate action that is a common sub-rule of the originals rules. A common sub-rule is modeled by applying the core rule of all additional actions (modeled by multi-rules).In execution; the multi-rules are relatively synchronized by core rule where a copy of core rule is embedded in each multi-rule. Consequently, the core rule runs only once. The embedding of multi-rules corresponds to the merged graph transformation [4].

Note there are may be an arbitrary number of multi- rules incorporating in the same core rule. Formally, the possibility of synchronization and integration of core rule in their multi- rules are defined by an interaction scheme. Note the formal structure of merged rules is described with the DPO approach (used by AGG) [3].

Interaction Scheme An interaction scheme IS= (r_k, M) consists of rule r_k called core rule and a set M= { r_i /1 ≤ i ≤ n} of rules called multi-rules with $r_k \subseteq r_i$ for all 1 ≤ i ≤n. All rules are typed over the same type graph.

Now, the above described example is executed with the merged graph transformation where the two transformation rules R_1 (***pay bill***) and R_2 (***settle bill***) have a common action "Ware Depend to Shop" is modeled by core rule. The first rule of Figure 7 shows the core rule followed by two multi-rules, that both incorporate the core rule and the actions that do not overlap. In the end, Figure 10 illustrates the merged rule. Therefore, the integration of parallel graph transformation based on merged rule into AGG can resolve the blocking problem.

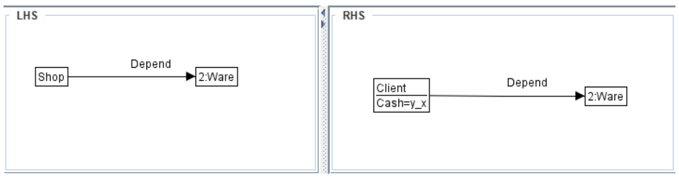


Figure 7. Critical pair

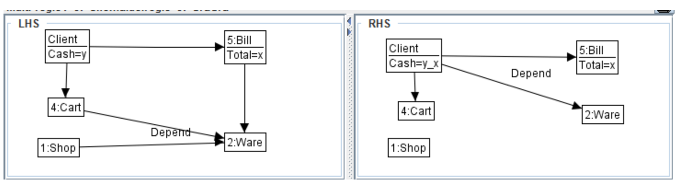


Figure 8. Multi rule1

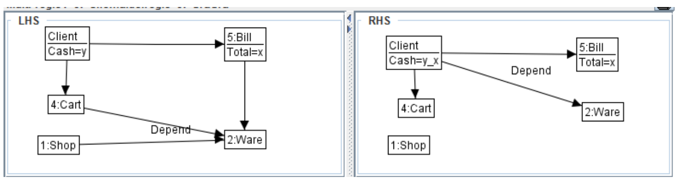


Figure 9. Multi rule2

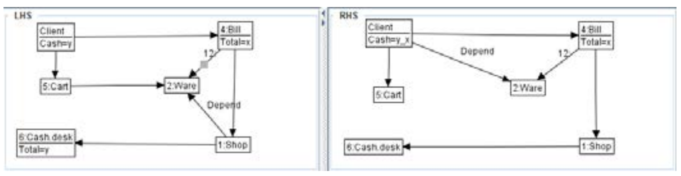


Figure 10. Merged rule

V. Related work

There are two transformation tools using parallel graph transformation: ATOM3 [8] and GROOVE [9]. AToM3 supports explicit definition of interaction types in different rule editors and GROOVE uses merge rule based on nested graph predicates.

Furthermore, graph transformation tool FuJaBA [10] uses so-called sets of nodes that are duplicated as often as necessary, but are not based on the transformation of merged graph.

A conceptual approach related to our approach, is parallel graph transformation for distributed graph states that has been studied in the framework of the algebraic theory of graph grammars [4]. Distributed graph transformation seems well adapted for asynchronous actions that operate completely independently.

VI. Conclusion

In this paper, we discussed parallel graph transformations where the concept of merging is very useful for this application domain because it permits to dependent transformation rules to be executed in parallel way.

The parallel actions that must be performed on a structure set of similar objects can be described by interaction types. A merged graph transformation applies an interaction type, i.e. a set of synchronized parallel actions relatively to the core rule. Although merged graph transformations are useful for specifying graph transformations more naturally and more efficiently, the theory is not fully developed. This work can be seen as an essential contribution to the merged graph transformation on AGG because practice results have shown that integrating merge approach in the AGG tool is efficient and runs smoothly without blocking in the case of independence between transformation rules.

Generalizing merged approach on the AGG tool in order to execute any parallel graph transformations, is feasible because AGG is “open source”.

References

[1] G. Rozsnberg, “Handbook of graph grammars and computing by graph transformation”, Vol 1. Foundations, Netherlands , 1996

[2] J. Bézivin, “Sur les principes de base de l’ingénierie des modèles”, RTSI- L’Objet, 2004, pp. 145-157.

[3] H.Ehrig, K.Ehrig et U.Prange,”Fundamentals of Algebrique Graph Transformation”, Monographs in Theoretical Computer Science. An EATCS Series, Springer, 2006.

[4] G .Taentzer,” Parallel and Distributed Graph Transformation: Formal Description and Application to Communication-Based Systems”, PhD thesis, TU Berlin 1996.

[5] D. D’Souza and A. Wills. Components and Frameworks with UML: The Catalysis Approach. Addison-Wesley, 1998.

[6] AGG Homepage, http://user.cs.tu-berlin.de/~gragra/agg/

[7] G,Taentzet,Towards Synchrnous and Asynchronous Graph Transformations, Fundamenta Informaticae, 1996.

[8] ATOM3 Homepage, http://atom3.cs.mcgill.ca/

[9] GROOVE Homepage, http://groove.cs.utwente.nl/

[10] FuJaBA Homepage, http://www2.cs.uni-paderborn.de/cs/agschaefer/Lehre/PG/FUJABA/