Supporting Software Agents by the Graph Transformation Systems

Leszek Kotulski

Institute of Computer Science Jagiellonian University
kotulski@ii.uj.edu.pl

Abstract. Agent systems demand from graph transformation systems not only an effectiveness of parsing but also supporting of the online derivation control (in order to visualize or control of their development). The way of the effective realization Derivation Control Environment for the distributed systems is presented. Finally, the computational effectiveness of the parsing, membership problems of the full graph transformation system is discussed.

1 Introduction

Multiagent systems consist of an agents environment and a number of software agents that are situated in that environment. Agents act autonomously, driven by their goals and plans, thereby sensing and reacting to their environment and cooperating with other agents. This environment represents not only hardware architecture but also the some logical resources (such as data bases, message subsystem, etc.) necessary to preform the agents task [9]. An individual agent can be executed in the computing unit if the proper local environment is present in this unit; otherwise either this agent should be moved to another computing unit or this unit environment has been enriched. Let us notice, that computing units (connected by communication media), resources associated with each environment and agent system can be represented, every separately, as a graph structures. The connection of the above structures is not only difficult to perform[1] but is also extremely difficult to visualize and manage.

The graph transformation mechanism seems to be useful in the support both the analysis some system properties and the online control of the distributed agent system description (DASD) modification. The first task was a matter of intensive and and finished with the success research [2]. These techniques are especially useful when we have a graph, which describes some problem, and we would like to answer if it belongs to the class of objects described by a given graph grammar. It is necessary to mark, that only few grammars can solve the membership problem with the polynomial computational complexity (as for example [6]). The graph transformations become also an important formal specification technique [3].

[1] We cannot simple join these graphs and trees because graph isomorphism in general case is an NP-complete problem.

V.N. Alexandrov et al. (Eds.): ICCS 2006, Part III, LNCS 3993, pp. 887–890, 2006.

The online control of the DASD modification needs the introduction of the derivation control mechanism, which is able online react on the events appearing in the distributed network. Such a proposition is presented in section two. Next, the usefulness of this proposition is illustrated by two examples: the coordination of changes in the agents environment and the optimization of the agent system effectiveness. For large systems it would be desirable if this control be held in the parallel way. Thus, at the end the parallel derivation and time complexity of the proposed solution is considered.

2 Derivation Control Environment

It is already proved that the graph transformations can be useful for problems describable by graphs, that properties should be checked. Problems appear when we need control dynamic changes occurring in the distributed environment as the reaction on an individual activity of its components. Unfortunately, individual components activity and their active cooperation are the key concept of the agent systems [1, 10].

We assume, that the minimal requirements to the graph transformation system useful for agent system seems to be following: react on-line to both user and software requests, solve a problem of the derivation control in a parallel way, be able to parse graphs with the polynomial computational complexity for grammars with sufficient descriptive power to solve problems at hand.

The last requirement can be fulfilled with an ETPL(k) graph grammar, used in such different areas as pattern recognition [6] and on-line allocation control [5]. The $O(n^2)$ computational complexity of parsing has been proved for this grammar [4]. To fulfill the rest of the above requests there is a need to design a special environment making possible an on-line parallel derivation control, which is called the **derivation control environment** (DCE). The basic term in this environment is the term **request**.

The **requests** are serviced by a DCE, that can be both monolithic and distributed one. A distributed DCE is a collection of monolithic DCEs, which communicate via system request. Formally DCE is introduced in [7]. Intuitionally (see Fig. 1), the derivation control diagram can be interpreted as a graph connecting **C**ontrol **P**oints (the dotted circles) inside which sequentially are evaluated the synchronizing function (if exists) and the selector choosing one of the transitions from the active **CP** to another one (drawn as an edge).

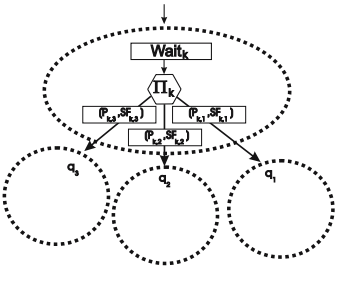

Fig. 1. Part of DCE

During such transition, both the production $P_{k,i}$ is applied and the semantics action $SF_{k,i}$ is executed. The semantics action SF (associated with the transition): enriches the order queue (requesting some

actions of the operating system), removes the **request** (that is serviced) from the request queue, evaluates parameters of the right-hand graph of the production P. The production P is applied to the current graph G creating a new graph G', in a way that is defined by transformation rules of the graph grammar associated with this derivation control diagram.

3 Derivation Control Environment and Agent Systems

Let us consider an example of usefulness of the graph transformation mechanism in the case of agent system migration problem. We assume that an agent can migrate in a distributed computing environment in order to find the least overloaded computing node. Unfortunately, the anomaly takes place when a new computing node is starting: all agents want to migrate to this node what immediately overload it. We assume that, DCE maintains the current allocation graph, and agent migration is associated with performing some productions updating this graph. The overloading problem can be solved if before migration each agent asks DCE for a permission [8]. This permission will be given only for a few agents. Moreover, such DCE can also be treated as an agent introducing the migration policy on the top level, by introducing some optimization algorithm. The ordered structure of the graph makes possible to complete such an optimization with the polynomial computational complexity of $(O(n^2))$ [7].

The second example is based on the problem pointed in [9]. We assume that a multiagent application consists of set of virtual machines (creating an local environment) and number of agents. The virtual machine enables for example the access to the local data base system by an agent. When an agent needs move to another destination it is limited only to this nodes which offers the same (or greater) virtual machine properties. Similarly as in the preceding example we assume that DCE maintains the allocation graph; its structure is however more complicated, there are represented there layers: hardware, virtual machines and agents. The role of DCE is in this case more complicated, and it can not be solved by the execution of two productions (removing agents node and putting it in another place). Application based on DCE have to check whether new virtual machine offers at least the same properties as the old one (it can be made using selectors). Otherwise, we have to enrich this environment before moving the agent to it. This task needs direct support of the graph grammar properties; as the old virtual machine is represented as a graph, so it is necessary to parse it in order to find a sequence of productions that can generate such a graph. These productions can be next performed in the context of a new computing node (creating the new virtual machine).

4 Conclusion

The centralized control of the agents allocation is the bottle-neck of the presented solution, thus it is desirable introduction the parallel derivation of the distributed graph. Let us assume that each monolithic DCE maintains the local

graph and some information about the connections of this graph nodes with nodes belonging to the local graphs that are maintained by other DCEs. Modifications inside local graphs can be made in a parallel way. It is, also, easy to coordinate the common modification of two local graphs by implementing some communication protocol (basing for example on Two Commit protocol) with help of their monolithic DCEs and some systems **requests**. The presented solution, for ETPL(k) graph grammar [4] associated with DCE , offers polynomial parsing and generation complexity ($O(n^2)$) (where n is maximum of local graph dimensions (n_i) and number of local DCEs (k)). In case of the AI solutions, using multiple agents, we can expect that the influence of k parameter on a final system effectiveness shall grow. The computational complexity is here specially outlined, because most of the graph transformation mechanisms with large description power has an unsatisfactory computational complexity. For example, the most popular graph transformation mechanism based on single push-out has exponential computational complexity of parsing. This excludes the use of such a methodology in applications working on-line.

References

1. Ralph Depke, Reiko Heckel, and Jochen Malte. Integrating visual modeling of agent-based and object-oriented systems. In *AGENTS '00: Proceedings of the fourth international conference on Autonomous agents*, pages 82–83, New York, NY, USA, 2000. ACM Press.
2. H. Ehrig, H.-J. Kreowski, U. Montanari, and G. Rozenberg, editors. *Handbook of graph grammars and computing by graph transformation: vol. 3: concurrency, parallelism, and distribution*. World Scientific Publishing Co., Inc., River Edge, NJ, USA, 1999.
3. H. Ehrig and G. Taentzer. Graphical represenation and graph transformation. *ACM Comput. Surv.*, 31(3es):9, 1999.
4. M. Flasiński. Power properties of nlc graph grammars with a polynomial membership problem. *Theor. Comput. Sci.*, 201(1-2):189–231, 1998.
5. M. Flasiński and L. Kotulski. On the use of graph grammars for the control of a distributed software allocation. *The Computer Journal*, 35:A167–A175, 1992.
6. Mariusz Flasiński. Characteristics of ednlc-graph grammar for syntactic pattern recognition. *Comput. Vision Graph. Image Process.*, 47(1):1–21, 1989.
7. L. Kotulski. *Model wspomagania generacji oprogramowania w rodowisku rozproszonym za pomoc a gramatyk grafowych*. Rozprawy Habilitacyjne. Wydawnictwo Uniwersytetu Jagiellońskiego, 2000. ISBN 83-233-1391-1.
8. L. Kotulski and A. Nowak. Wykorzystanie gramatyk grafowych do kontroli migracji mobilnych agentów. In *Inżynieria wiedzy i systemy ekspertowe*, pages 319–325. Oficyna wydawnicza Politechniki Wroclawskiej, 2003.
9. Koenraad Mertens, Tom Holvoet, and Yolande Berbers. A case for adaptation of the distributed environment layout in multiagent applications. In *SELMAS '05: Proceedings of the fourth international workshop on Software engineering for large-scale multi-agent systems*, pages 1–8, New York, NY, USA, 2005. ACM Press.
10. Rong Xie, Daniela Rus, and Cliff Stein. Scheduling multi-task multi-agent systems. In *AGENTS '01: Proceedings of the fifth international conference on Autonomous agents*, pages 159–160, New York, NY, USA, 2001. ACM Press.