A Translation from Object-Based Hypergraph Grammars into π-Calculus[*]

Luciana Foss[1] **, Leila Ribeiro**[1]

[1]Instituto de Informática – Universidade Federal do Rio Grande do Sul
Caixa Postal 15064 – 90501-970 Porto Alegre, RS

lfoss@inf.ufrgs.br,leila@inf.ufrgs.br

Abstract. *Object-based models offer abstract constructions to describe complex systems. The Object-Based Graph Grammar (OBGG) is a very intuitive formalism that may be used to describe this kind of systems. To be really useful in practice, there should be a (preferably automatic) way to verify whether the desired properties of a system are fulfilled by the model constructed using graph grammars. However, up to now, there are no automatic tools for verification of OBGGs. In this work we propose a translation from Object-Based Graph Grammars into* π*-Calculus. So, we may be able to prove properties of systems modeled in OBGGs using existing automatic checkers for* π*-calculus.*

1. Introduction

One of the main aims of rigorous software development is to assure the correctness of the developed system. The basis of a rigorous development is the use of a formal specification method, with syntax and semantics well defined. There are several formalisms for specification of computational systems and the choice of which one to use depends on the characteristics of the application to be developed. Object-based models offer an abstraction level that has been successfully applied in practice, where operations and data are described together within one object. Object-based models are specially well-suited to the specification of concurrent and cooperating systems. A formal object-based specification language was proposed in [Dotti and Ribeiro, 2000]: Object-Based Graph Grammars (OBGG). OBGGs are a restricted form of graph grammars [Ehrig, 1978] that, besides the features of object-based languages, offer a visual specification language, which is usually welcomed by practitioners. However, to be really useful in practice, there should be a (preferably automatic) way to verify whether the desired properties of a system are fulfilled by the model constructed using graph grammars. Currently, models defined in this formal specification language can be analyzed through simulation [Copstein et al., 2000]. It is also possible to generate code for execution in a real environment, following a straightforward mapping from an OBGG model to code [Duarte, 2001]. Up to now, there are no automatic tools for verification of OBGGs. Instead of constructing such tools from scratch, an alternative way is to define a (semantics preserving) translation from this specification language into another for which automatic verification tools already exist. This is what we will do in this thesis.

[*]This work was supported by CNPq, FAPERGS (project ForMOS) and CNPq/CNR.

The π-calculus [Milner and Parrow, 1992] is a well known and established formalism for description of semantics of concurrent systems. There are some automatic checkers for this formalism, for example, HAL [Ferrari et al., 1998] and MWB (Mobility Workbench) [Victor and Moller, 1994]. Although one may use the π-calculus to write specifications, its original intention was to serve as a semantic model language. And indeed, specifications of practical applications written in π-calculus tend to become large and cumbersome.

In this work we define a translation from OBGG into π-calculus that is a first step to join the advantages of both methods: the visual, intuitive, and object-based style of graph grammars and the verification tools and semantical model of π-calculus. Moreover, we prove that this translation preserves the semantics of OBGGs.

2. Object-Based Hypergraph Grammars - OBHG

Graph grammars offer a natural way to express complex situations. The system states are described by graphs and the dynamic aspects may be captured by grammar rules. We use, in this work, a model based on Object-Based Graph Grammar (OBGG) [Dotti and Ribeiro, 2000], called Object-Based Hypergraph Grammars (OBHG). In this model, objects and messages are represented by vertices and hyperedges, respectively. Each hyperedge has one target (the target of the message) and zero or more sources (the parameters of the message). The internal state of an object will not be considered, that is, the objects do not have attributes. An extended version of this work including attributes is currently under development.

Object reactions to the receiption of a message will be modeled by grammar rules. Each rule describes the processing of only one message, that may result in the creation of vertices (objects) and/or hyperedges (messages). The hyperedge corresponding to the processed message is deleted with the rule application. We may have more than one rule processing the same kind of message, where the choice of the rule to be applied is non-deterministic, modeling non-deterministic operations. The concurrency between objects and the internal concurrency are modeled by parallel application of rules. We may have different types of objects and messages in a system. In this work, we use a type (hyper)graph, called type (hyper)graph, to specify the type of each system element. An OBHG consists of a type (hyper)graph, an initial (hyper)graph, that describes a initial state of the system, and a set of rules.

In figure 1 we show an example of OBHG, where T is a type graph, H is a initial graph and r is the only rule of this OBHG. T describes all possible objects and messages of the system. The objects are represented by squares and ellipses, and the messages are represented by pentagons together with its links. In T we can see two kinds of objects: Car and pos, and one kind of message: $next$. The messages $next$ have as target an pos and three parameters: two pos and one Car. In the initial graph (H), the objects and messages have an index to distinguish the several instances of each object and message. The rule describes the behavior of an pos object when it receives a $next$ message. This rule consumes a $next$ message (left-hand side) end creates a new $next$ message (right-hand side). The same rule can be applied twice (in parallel) to messages $next_1$ and $next_2$.

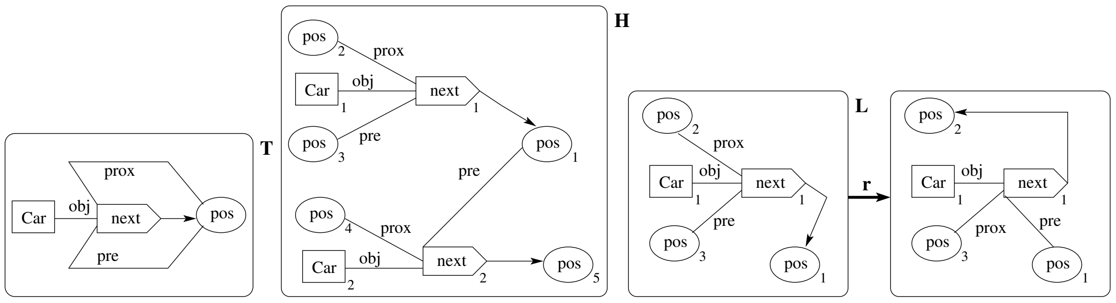

Figure 1: Example of an OBHG.

3. π-Calculus

The π-calculus [Milner and Parrow, 1992] is a process algebra that handles channels as messages, thus modeling processes that may have changing structure. The basic computational step is the sending of a channel between two processes. The receiver process may use the new channel in future interactions. The channels in the π-calculus are only names, atomic entities without internal structure. A process can be composed by agents. These agents are abstractions of processes, that is, they are identifiers that will be replaced by a process. Each agent identifier must be defined, i.e., we must describe the process it represents. A system specification describes the initial state of this system. In π-calculus, this is done through a term (process). In this specification, all agent identifiers must be defined. The state changes of a system are described by term rewriting. The term rewriting is defined by some transition rules, that describe the way an agent changes after an interaction.

An example of a π-calculus process is $(\nu x, y)(A(x, y)|B(y, x))$. In this process, there are two agents, A and B, executing in parallel. The names x and y are new names in this process, and they are the parameters of the agents (they will be private channels used by these two agents). The agent definitions are: $A(t, s) = \bar{t}s.0$ and $B(s, t) = t(s).0$. The behavior of this process is the sending of the name s from agent A to agent B through channel t (the processes synchronize on this channel).

4. Translation from OBHG into OBM-π

The translation proposed in this work is the translation from objects, messages and their relationships into agents of the π-calculus. These target agents have specific forms. The kind of π-calculus terms that characterize the translated graph grammars are called Object-Based Model described in π-Calculus, short OBM-π.

In an OBM-π, the objects and messages are defined as processes of the π-calculus, called object agents and message agents, respectively. These agents communicate through a local channel. The source object and the parameters of each message are represented as parameters of message agents. The reactions of the objects when receiving a message are described by rule agents, that compose the object agent. Each kind of message can have several procedures for treating it, so we can have several rule agents that describe the treatment of the same kind of message. The choice of the procedure to be executed is non-deterministic. The concurrency between objects is modeled by parallel composition of object agents and message agents. So each object can treat its messages in parallel. The internal concurrency is modeled by recursion of the object agent.

4.1. Semantic Compatibility

In order to compare the semantics of the OBHG with the one of π-calculus, we defined an interleaving semantics for OBHG, because this is the usual semantic model of π-calculus. In this model, non-determinism is described by different sequences of computations with common prefix. The concurrency is modeled by different computation steps observed in two different orders.

The synchronizations performed by agents simulate the grammar rule applications. The treatment of one message, in an OBHG, is done applying only one rule, thus the OBHG transition system presents only one transition for each treated message. On the other hand, in an OBM-π, the treatment of one message requires various synchronizations between object and message agents. Thus, the OBM-π transition system has a sequence of transitions for each message. In an OBM-π, the resulting synchronizations of parallel treatments of two messages create some paths where the transitions (of treatment of each message) may appear interleaved. In an OBHG the treatment of messages is atomic. In the LTS of the translation of this OBHG there exist paths that represent these atomic treatments. These paths are called **complete paths**. The others paths do not represent existenting behavior in OBHG. So, the semantic compatibility is given only for paths in OBM-π LTS that represent the atomic treatment of messages, that is, for all complete paths. The effect of selecting these paths is discussed in the next section. To compare the semantics of both models we remove the silent transitions and translate the valid terms into typed hypergraphs. To prove that the translation preserves the semantics of OBHG we showed that all paths that belong to OBHG LTS also belong to the set of complete paths of its translation and vice-versa.

5. Conclusion and Future Work

In this work we have proposed a translation from object-based hypergraph grammars into π-calculus agents, more specifically, into agents that have a specific form described by the OBM-π model. We compared the LTSs of the two models to prove that the translation preserves the OBHG semantics. However, there is a change of granularity in the translation: an action that was performed atomically in OBHG (one rule application) is performed in many steps when we consider its translation into π-calculus. This means that not all sequences of transitions of a translated OBHG correspond to computations of the original OBHG. Therefore, if we use the translated model to prove properties, we should take care about the results. If a property is satisfied, then it is also satisfied by the original grammar (because all paths of the OBHG LTS are included in the LTS of the translated OBHG). If a property is not satisfied, it does not necessarily means that the property does not hold for the original OBHG because it could be that it fails for one of the paths of the translated OBHG that does not correspond to a path of the original OBHG. But, as the observable labels of the two LTSs are exactly the same (and occur in the same orders), we conjecture that many of the properties in which we are interested in (that properties involving these labels) are also preserved by the translation. This is currently under investigation.

If one wants to use π-calculus tools to verify properties of the systems specified using OBHGs, the user must know the language of specification of properties used by that tool, for example π-logics. The user does not actually need to know how the translation was done, because the labels of the transition systems, that are the events over which we can define properties, are the same in both transition systems (and the occurrences of these events are in the same orders). However, it could be useful to have a graphical logical language to describe the properties that is closer to the specification language OBHGs.

In this work we did a first step towards automatic verification of OBHGs, namely, we defined a translation from OBHGs into the input language of some verification tools (π-calculus). To really provide OBHGs with automatic verification mechanisms, this translation shall be implemented. We stress the fact that, in contrast to other approaches of translating specification languages into input languages of verification tools, we provided a complete proof of the preservation of the semantics of our translation. This is a highly complex task, because it involves deep knowledge of the semantical models of the specification and target (verification) language, as well as good theoretical skills to manipulate these models. However, once this proof is done, the user of our translation does not have to know anything neither about how the translation was defined, nor about how the proof was carried out. Moreover, he/she can be completely sure that the properties proved by the verification tool over the translated model really hold for his original OBHG specification.

References

Copstein, B., Móra, M. d. C., and Ribeiro, L. (2000). An Environment for Formal Modeling and Simulation of Control Systems. In *Proc. 33rd Annual Simulation Symposium*, pages 74–82, Washington. Institute of Electrical and Electronics Engineers.

Dotti, F. L. and Ribeiro, L. (2000). Specification of Mobile Code Systems Using Graph Grammars. In *4 IFIF TC6 WG6.1 International Conference on Formal Methods for Open Object-Based Distributed Systems*, pages 45–63, Stanford. Boston: Kluwer Academic.

Duarte, L. (2001). Development of distributed systems with mobile code through formal specification (in portuguese). Master's thesis, Pontifícia Universicade Católica do Rio Grande do Sul.

Ehrig, H. (1978). Introduction to the Algebraic Theory of Graph Grammars. In *Proc. International Workshop on Graph-Grammars and Their Application to Computer Science and Biology*, volume 73 of *Lecture Notes in Computer Science*, pages 1–69, Bad Honnef. Berlin: Springer Verlag.

Ferrari, G., Gnesi, S., Montanari, U., Pistore, M., and G.Ristori (1998). Verifying Mobile Processes in the HAL Environment. In *International Conference on Computer Aided Verification*, volume 1427 of *Lecture Notes in Computer Science*, pages 511–515, Vancouver, CA. Berlin: Springer Verlag.

Milner, R. and Parrow, J. (1992). A Calculus for Mobile Processes I and II. *Information and Computation*, 100:1–77.

Victor, B. and Moller, F. (1994). The Mobility Workbench - a tool for the π-calculus. In Dill, D., editor, *Proc. International Conference on Computer Aided Verification, CAV*, volume 818 of *Lecture Notes in Computer Science*, pages 428–440. Berlin: Springer-Verlag.