On the Concurrent Semantics of Algebraic Graph Grammars*

Paolo Baldan[1] and Andrea Corradini[2]

[1] Dipartimento di Informatica, Università Ca' Foscari di Venezia, Italy
baldan@dsi.unive.it
[2] Dipartimento di Informatica, Università di Pisa, Italy
andrea@di.unipi.it

Abstract. Graph grammars are a powerful model of concurrent and distributed systems which can be seen as a proper extension of Petri nets. Inspired by this correspondence, a truly concurrent semantics has been developed along the years for the algebraic approaches to graph grammars, based on Winskel's style unfolding constructions as well as on suitable notions of processes. A basic role is played in this framework by the study of contextual and inhibitor nets, two extensions of ordinary nets which can be seen as intermediate models between ordinary Petri nets and algebraic graph grammars.
This paper presents a survey of these results, discussing in a precise, even if informal way, some of the main technical contributions that made possible the development of such a theory.

Introduction

Petri nets [40, 42] are one of the most widely used models of concurrency. Since their introduction they have attracted the interest of both theoreticians and practitioners. Along the years Petri nets have been equipped with satisfactory semantics, doing justice to their intrinsically concurrent nature. These semantics have served as basis for the development of a variety of modelling and verification techniques. However, the simplicity of Petri nets, which is one of the reasons of their success, represents also a limit in their expressiveness. If one is interested in giving a more structured description of the state, or if the kind of dependencies between steps of computation cannot be reduced simply to causality and conflict, Petri nets are likely to be inadequate.

This paper summarizes the work presented by the authors in a series of papers [2–4, 7–10, 12], most of which written jointly with Ugo Montanari, and some with Nadia Busi, Michele Pinna and Leila Ribeiro. Such papers are the outcome of a project aimed at proposing graph transformation systems as an alternative model of concurrency, extending Petri nets. The basic intuition underlying the use of graph transformation systems for formal specifications is to represent the

* Research partially supported by the EU FET-GC Project IST-2001-32747 AGILE, and by the EC RTN 2-2001-00346 SEGRAVIS.

H.-J. Kreowski et al. (Eds.): Formal Methods (Ehrig Festschrift), LNCS 3393, pp. 3–23, 2005.

states of a system as graphs (possibly attributed with data-values) and state transformations by means of rule-based graph transformations. Needless to say, the idea of representing system states by means of graphs is pervasive in computer science. Whenever one is interested in giving an explicit representation of the interconnections, or more generally of the relationships among the various components of a system, a natural solution is to use (possibly hierarchical and attributed) graphs. The possibility of giving a suggestive pictorial representation of graphical states makes them adequate for the description of the meaning of a system specification, even to a non-technical audience. A popular example of graph-based specification language is given by the Unified Modelling Language (UML), but we recall also the more classical Entity/Relationship (ER) approach, or Statecharts, a specification language suited for reactive systems. Moreover, graphs provide a privileged representation of systems consisting of a set of processes communicating through ports.

When one is interested in modelling the *dynamic aspects* of systems whose states have a graphical nature, graph transformation systems are clearly one of the most natural choices. Since a graph rewriting rule has only a local effect on the state, it is natural to allow for the parallel application of rules acting on independent parts of the state, so that a notion of concurrent computation naturally emerges in this context. The research in the field, mainly that dealing with the so-called *algebraic approaches* to graph transformation [25, 22, 27], has led to the attempt of equipping graph grammars with a satisfactory semantical framework, where their truly concurrent behaviour can be suitably described and analyzed. After the seminal work [31], which introduced the notion of *shift equivalence*, many original contributions to the theory of concurrency for algebraic graph transformation systems have been proposed during the last ten years, most of them inspired by their relation with Petri nets. In particular, for the *double-pushout* (DPO) approach to graph transformation, building on some ideas of [31], a *trace semantics* has been proposed in [18, 22]. Resorting to a construction in the style of Mazurkiewicz, the trace semantics has been used to derive an *event structure semantics* [20, 19] for DPO graph grammars. Graph grammars have been endowed also with a process semantics with the introduction of *graph processes* [21], further refined with the notion of concatenable (deterministic) processes [8]. A Winskel's style unfolding construction [51] has been defined both for the *single pushout* (SPO) and the DPO approaches [43, 9, 10], and has been exploited for providing, through suitable chains of functors, such grammars with more abstract semantics based on event structures and domains.

In this survey paper, after recalling the basics of the algebraic approaches to graph transformation and their relationship with Petri nets, we will summarize the functorial, unfolding semantics of Petri nets and the elegant way in which it can be reconciled with the event structure semantics based on deterministic processes. Next we will discuss how this approach has been generalized to algebraic graph grammars. This required the definition of new structures and constructions that will be briefly outlined in the following sections. We shall focus mainly on the definition of two generalizations of prime event structures, called