Graph Processes with Fusions: Concurrency by Colimits, Again*

Fabio Gadducci and Ugo Montanari

Dipartimento di Informatica, Università di Pisa,
Pisa, Italy
{gadducci,ugo}@di.unipi.it

Abstract. Classical concurrency in the DPO approach to graph rewriting, as defined by the shift equivalence construction [7], can also be represented by a graph process, a structure where concurrency and causal dependency are synthetically represented by a partial ordering of rewrites [1]. Interestingly, all shift equivalent derivations, considered as diagrams in the category of graphs, have the same colimit, which moreover exactly corresponds to the graph process. This construction, due to Corradini, Montanari and Rossi, was originally defined for rules with injective right-hand morphisms [6]. This condition turns out to be restrictive when graphs are used for modeling process calculi like ambients [4] or fusion [21], where the coalescing of read-only items is essential [11, 13]. Recently, a paper by Habel, Müller and Plump [16] considered again shift equivalence, extending classical results to non-injective rules. In this paper we look at the graph-process-via-colimit approach: We propose and motivate its extension to non-injective rules in terms of existing computational models, and compare it with the aforementioned results.

Keywords: DPO rewriting, concurrent semantics, process calculi.

1 Introduction

Historically, graph rewriting lies its roots on the late Sixties, as the conceptual extension of the theory of formal languages: The extension was motivated by a wide range of interests, from pattern recognition to data type specification. Nowadays, the emphasis has shifted from the generative aspects of the formalism, moving toward what could be called the "state transformation" view: A graph is considered as a data structure, on which a set of rewriting rules may implement local changes; the transformation mechanism itself is considered as expressing a basic computational paradigm, where graphs describe the states of an abstract machine and rewrites express its possible evolutions. An interest confirmed by the large diffusion of visual specification languages, such as the standard UML, and the use of graphical tools for their manipulation.

To some extent, this is also the intuition behind the introduction of process algebras, such as Milner's CCS [19]: They represent specification languages for

* Research partially supported by the EU within the FET – Global Computing Initiative, project AGILE IST-2001-32747 (*Architectures for Mobility*).

H.-J. Kreowski et al. (Eds.): Formal Methods (Ehrig Festschrift), LNCS 3393, pp. 84–100, 2005.

concurrent systems, considered as structured entities interacting *via* some synchronization mechanism. A (possibly distributed) system is just a term over a signature, under the hypothesis that each operator represents a basic feature of the system. The rewriting mechanism (accounting for the interaction between distinct components of a system) is usually described operationally, according to the SOS-style [22], where the rewriting steps are inductively defined by inference rules, driven by the structure of terms. Novel extensions of the process algebra paradigm involve calculi with higher-order features such as name mobility (hence, *nominal calculi*). Here systems are terms, carrying a set of associated *names*, and usually provided with a *structural* congruence, expressing basic observational properties; the reduction mechanism may also change the topology of a system, which formally amounts to changing the associated set of names.

Recent years have seen many proposals concerning the use of graph rewriting techniques for simulating reduction in process algebras, in particular for their mobile extensions. Typically, the use of graphs allows for getting rid of the problems concerning the implementation of reduction over the structural congruence, such as e.g. the α-conversion of (bound) names, since equivalent processes turn out to be mapped into isomorphic graphs. Most of these proposals follow the same pattern: At first, a suitable graphical syntax is introduced, and its operators used for implementing processes. After that, usually ad-hoc graph rewriting techniques allows for simulating the reduction semantics. Most often, the resulting graphical structures are eminently hierarchical (that is, roughly, each node/edge is a structured entity, and possibly a graph). From a practical point of view, this is unfortunate, since the restriction to standard graphs would allows for the reuse of already existing theoretical techniques and practical tools.

Building on our work on the syntactical presentation of rule-based graphical formalisms [3, 5, 12] (using techniques adopted in the algebraic specification community for modelling flow graphs [8]), in recent years we proposed graphical encodings of (possibly recursive) processes of π-calculus [20] and mobile ambients [4] into unstructured graphs, proving suitable soundness and completeness results with respect to the original reduction semantics (see [11] and [13], respectively, also for a comparison with other proposals for the graphical encoding of calculi with name mobility). The use of non hierarchical graphs allows for the reuse of standard graph rewriting theory and tools for simulating the reduction semantics of these calculi, such as the double-pushout (DPO) approach. (A more specific discussion on the advantages offered by our graphical encodings appears in the last paragraph of Section 5.3 of the present paper.)

Having asserted the benefits concerning the encoding of nominal calculi into unstructured graphs and the use of the DPO approach, we face some unresolved problems with respect to the concurrent semantics. The correspondence between graph transformation and process reduction highlights the relevance of concurrency in this setting, since allowing for the simultaneous execution of independent rewrites implicitly defines a concurrent semantics for process reduction.

Unfortunately, the graphical encodings we proposed so far (including that for the simple calculus in Section 5 of the present paper) lies outside the canon