Available online at www.sciencedirect.com
ScienceDirect
Electronic Notes in Theoretical Computer Science
ELSEVIER
Electronic Notes in Theoretical Computer Science 72 (2007) 45–58
www.elsevier.com/locate/entcs

# A Higher-Order Calculus for Graph Transformation[1]

Maribel Fernández and Ian Mackie[2,3]

*Department of Computer Science, King's College, Strand, London WC2R 2LS*

Jorge Sousa Pinto[4]

*Departamento de Informática, Universidade do Minho, 4710-057 Braga, Portugal*

**Abstract**

This paper presents a formalism for defining higher-order systems based on the notion of graph transformation (by rewriting or interaction). The syntax is inspired by the Combinatory Reduction Systems of Klop. The rewrite rules can be used to define first-order systems, such as graph or term-graph rewriting systems, Lafont's interaction nets, the interaction systems of Asperti and Laneve, the non-deterministic nets of Alexiev, or a process calculus. They can also be used to specify higher-order systems such as hierarchical graphs and proof nets of Linear Logic, or to specify the operational semantics of graph-based languages.

*Keywords:* Higher-order system, graph transformation, Combinatory Reduction System, graph rewriting system

## 1 Introduction

Rule-based transformations of graphs have been used in many areas of computer science, including the specification and development of software systems, the definition of visual languages, the implementation of programming languages (see [5,25]). The notion of interaction, which can be seen as a particular kind of graph transformation, has been used to model concurrent systems [23], to give a semantics to (linear) logic proofs [11], as a programming discipline [17], and as an implementation technique for functional languages [3]. In each case, a syntax and an operational semantics (a calculus) has been defined, often independently.

[1] Partially supported by TMR LINEAR.

[2] Email: maribel@dcs.kcl.ac.uk

[3] Email: ian@dcs.kcl.ac.uk

[4] Email: jsp@di.uminho.pt

1571-0661 © 2007 Elsevier B.V. Open access under CC BY-NC-ND license.
doi:10.1016/j.entcs.2002.09.005

In this paper we present a higher-order language that can serve to specify interaction systems as well as graph and term-graph rewriting systems. The syntax is inspired by the Combinatory Reduction Systems (CRSs) of Klop [16], and can be seen as a generalization of the equational notation for term-graph rewriting [2]. We demonstrate its use by giving several examples of application, including the definition of hierarchical graphs (where it is possible to abstract subgraphs, see [4] for more details), a first-order interaction language together with its operational semantics (all in the same language), and the specification of higher-order program transformations and optimization schemes. The latter will be defined for Lafont's interaction nets [17].

¿From a practical point of view, the higher-order syntax can be used as a tool in the design and implementation of graphical languages: it allows us to express not only graphical programs but also their operational semantics (including evaluation strategies and optimization schemes), type systems, and transformations used in the proof of meta-theoretical properties of programs. An instance of the latter kind of transformation is the *packing operator* defined by Lafont [19] to prove the universality of the interaction combinators (a specific system of interaction nets in which every interaction net can be encoded). Other packing and unpacking operations have been described in [9], and they can all be formally defined using higher-order rules in our system.

Another aspect where the higher-order syntax presents advantages is for structuring and modularizing programs defined by graphs (or nets). Hierarchical definitions are very useful in the framework of graph rewriting [4], and the same techniques can be exported to interaction nets using the higher-order syntax. In particular, the operation that combines two interaction nets to produce a new net where one or more edges have been connected together (the analogous of application in functional programming) is currently a meta-operation. We show how to internalize it using the higher-order language, and give examples where this technique is used to write modular programs. Once we have the ability to model the combination of nets, it is straightforward to express a notion of higher-order interaction nets, where a net depends on another net. As with functional programming, this technique can be used to write recursive nets: nets which depend on themselves.

*Related Work.* Our syntax is inspired by CRSs, but similar results can be obtained by using other higher-order systems, such as Nipkow's Higher-order Rewrite Systems [22], or Khasidashvili's Expression Reduction Systems [13]. The three formalisms are closely related [26]. CRSs have been used in previous work on interaction nets: Laneve [20] defined Interaction Systems as CRSs, and in [7] a translation function is given from interaction nets to CRSs.

Van Raamsdonk [27] defines a class of higher-order rewrite systems with a general notion of substitution and shows the encoding of several languages, including Interaction Systems and Proof Nets of linear logic. Our goal is more specific: our higher-order textual notation has been designed to represent graph-based transformations, and therefore the calculus contains specific graph-oriented features. The notation used to represent graphs in the calculus is a generalization of the equational

Download English Version:

https://daneshyari.com/en/article/424343

Download Persian Version:

https://daneshyari.com/article/424343

Daneshyari.com