Science of Computer Programming 94 (2014) 255–288

Contents lists available at ScienceDirect

Science of Computer Programming

www.elsevier.com/locate/scico

A sound and complete theory of graph transformations for service programming with sessions and pipelines

Liang Zhao [a,*], Roberto Bruni [b], Zhiming Liu [c]

[a] Institute of Computing Theory and Technology, Xidian University, 2 South Taibai Road, Xi'an 710071, China
[b] Department of Computer Science, University of Pisa, Largo B. Pontecorvo 3, I-56127 Pisa, Italy
[c] Software Engineering Group, School of Computing, Telecommunication and Networks, Birmingham City University, Birmingham B4 7XG, UK

HIGHLIGHTS

- Algebra of hierarchical graphs that supports Double-Pushout graph transformations.
- Representation of service systems as hierarchical graphs.
- Representation of behaviors of service systems as graph transformation rules.
- Soundness and completeness of graph transformation rules.

ARTICLE INFO

Article history:
Received 31 October 2011
Received in revised form 16 August 2013
Accepted 11 November 2013
Available online 27 November 2013

Keywords:
Process calculus
Hierarchical graph
Graph transformation

ABSTRACT

Graph transformation techniques, the Double-Pushout (DPO) approach in particular, have been successfully applied in the modeling of concurrent systems. In this area, a research thread has addressed the definition of concurrent semantics for process calculi. In this paper, we propose a theory of graph transformations for service programming with sophisticated features such as sessions and pipelines. Through graph representation of CaSPiS, a recently proposed process calculus, we show how graph transformations can cope with advanced features of service-oriented computing, such as several logical notions of scoping together with the interplay between linking and containment. We first exploit a graph algebra and set up a graph model that supports graph transformations in the DPO approach. Then, we show how to represent CaSPiS processes as hierarchical graphs in the graph model and their behaviors as graph transformation rules. Finally, we provide the soundness and completeness results of these rules with respect to the reduction semantics of CaSPiS.

1. Introduction

Process calculi are a flexible mathematical formalism that provides a convenient abstraction for concurrent systems, in the same way as λ-calculus lays the foundation of sequential computation. A process calculus generally has two main ingredients: an algebra of computational entities, including a set of structural congruence axioms, and an operational semantics for modeling the evolution of processes. The entities are called *processes* and they are constructed by primitives such as communication and parallel composition. The operational semantics is provided either in terms of a labeled transition system, or as a reduction system that poses the basis for studying several notions of behavioral equivalence over processes.

* Corresponding author. Tel./fax: +86 29 88202883.
E-mail addresses: lzhao@xidian.edu.cn (L. Zhao), bruni@di.unipi.it (R. Bruni), zhiming.liu@bcu.ac.uk (Z. Liu).

0167-6423/$ – see front matter
http://dx.doi.org/10.1016/j.scico.2013.11.029

Process calculi have become quite mature in the study of traditional concurrent and communicating systems [1,2], and even advanced to specification and verification of mobile systems [3]. However, the traditional process calculi do not match certain advanced features of service-oriented computing (SOC) like the nested scoping of sessions or pipelining workflows, as well as the interplay between linking and containment. Though there are attempts of using π-calculus [3] as a model of service systems [4,5], the modeling is of quite low level and different first-class aspects in SOC, such as client-service interaction and orchestration, are mixed up and obfuscated. The low level communication primitives of π-calculus make the analysis quite complicated. Particularly, with the same communication pattern used to encode many different aspects, it is almost infeasible to re-use static analysis techniques to provide any guarantees about safe interactions.

In order to improve this situation, a few service-oriented calculi are proposed. Service Centered Calculus (SCC) [6] introduces *service definition*, *service invocation* and *session handling* as first class modeling elements, so as to model service systems at a better level of abstraction. However, SCC has only a rudimentary mechanism for handling session closure, and it has no mechanism for orchestrating values arising from different activities. These aspects are improved in *Calculus of Session and Pipelines* (CaSPiS) [7]. CaSPiS supports most of the key features of SCC, but the notions of session and pipelining play a more central role. A *session* has two sides (or participating processes) and it is equipped with protocols followed by each side during an interaction between the two sides. A *pipeline* permits orchestrating the flow of data produced by different sessions. The concept of pipeline is inspired by Orc [8], a basic and elegant programming model for orchestration of computation entities. A structured operational semantics of CaSPiS is given in [7] based on labeled transitions. It does yet have a simpler and compact reduction semantics [9], on which we focus, that deals with silent actions of processes in the labeled transition system. In [9], the relation of CaSPiS and π-calculus in terms of their mutual embeddings is also discussed.

As illustrated by a large body of literature, graphs and graph transformations provide useful insights into distributed, concurrent and mobile systems [10–12]. Following this direction, we are going to define a graph-based concurrent semantics for CaSPiS. This can help, for example, to record causal dependencies between interactions and exploit such information for detecting the possible source of faults and misbehaviors. For this, we need to address two issues. The first is that sessions and pipelines introduce a strong hierarchical nature into a service-oriented system in both of its static structure and dynamic behavior. The hierarchical structure also changes during the evolution of the system, due to dynamic creation of nested sessions through invocation of services, and dynamic creation of processes with execution of pipelines. Therefore we must deal with hierarchical graphs. The second is that the graph transformation semantics must be "compatible" with the existing interleaving one.

In this paper, we propose a hierarchical graph representation of service systems specified by CaSPiS and show how to use graph transformation rules to characterize their behaviors. More precisely, our first contribution is to set up a model of hierarchical graphs by exploiting a suitable graph algebra. In this model, graph transformations are studied following the well-studied Double-Pushout (DPO) approach [13]. Then, we map CaSPiS processes to hierarchical graphs in the graph algebra and define a graph transformation system with a few sets of graph transformation rules. As a main result of this paper, we proved that the graph transformation system is not only sound, but also complete with respect to the reduction semantics of CaSPiS.

This paper extends the previous workshop version [14] in several aspects. In this paper, we modeled more sophisticated features of CaSPiS including replication processes and constructed values. Such an extension improves the expressiveness of our framework, enabling us, for example, to reason about persistent services that are always available for invocations. Another progress made in this paper is the provision of a full graph transformation framework of reduction, including rules for process copy, data assignment and garbage collection. Furthermore, this paper provides the proof of soundness and completeness of the whole graph transformation system.

There are various models of graphs and graph transformations that aim at characterizing and visualizing distributed, concurrent and mobile systems, including service systems. Gadducci proposes a graphical implementation of π-calculus in [10] based on term graph rewriting [15,16]. In this work, processes of π-calculus, including recursive ones, are encoded into term graphs, which are directed and acyclic hypergraphs over a chosen signature representing "terms with shared sub-terms" over the signature. The use of term graphs makes it straightforward to re-use the standard graph rewriting technique, such as the DPO approach, which leads to a non-deterministic concurrent semantics. Then, the soundness and completeness of the encoding is verified by proving the equivalence of the concurrent semantics and the original reduction semantics of π-calculus. Milner provides a behavior semantics for condition-event Petri nets [17] in [18] based on bigraphs and their reactive systems [19,11]. Generally, a bigraph consists of two orthogonal structures: a place graph and a link graph representing locality and connectivity of agents, respectively. In this work, a condition-event Petri net is modeled as a bigraph whose place graph is flat, and then the behavior of the net is modeled as a bigraphical reactive system equipped with a labeled transition system and an associated bisimilarity equivalence. This bisimilarity is shown to coincide with the original one of condition-event Petri nets. Another graph-based framework is presented by Hirsch and Tuosto [20] for specifying systems with high-level Quality of Service (QoS) aspects, where constraint-semirings [21] are used to describe QoS requirements of various criteria. The framework is based on Synchronized Hyperedge Replacement (SHR) [22,12], a hypergraph rewriting mechanism for modeling the reconfiguration of (possibly distributed) systems. In SHR, the behavior of a single edge is defined by the notion of production, which indicates how and under what condition an edge can be replaced by a generic graph. Then, global transitions are obtained by synchronizing applications of productions with compatible