Event Structure Semantics of Orc*

Sidney Rosario¹, David Kitchin³, Albert Benveniste¹, William Cook³, Stefan Haar⁴, and Claude Jard²

¹ Irisa/Inria, Campus de Beaulieu, 35042 Rennes cedex, France
² Irisa/ENS Cachan, Campus de Beaulieu, 35042 Rennes cedex, France
³ The University of Texas at Austin, Department of Computer Sciences, Austin, USA
⁴ Irisa/Inria Rennes and SITE, University of Ottawa, Canada

Abstract. Developing wide-area distributed applications requires jointly analyzing functional and Quality of Service (QoS) aspects, such as timing properties. Labelled transition systems and sequential trace semantics - the common semantic domains - do not facilitate this kind of analysis because they do not precisely express the causal relationships between events. Asymmetric Event Structures (AES) provide an explicit representation of the causal dependencies between events in the execution of a system and allow for an elegant coding of preemption. Event structures are, however, difficult to construct compositionally, because they cannot easily represent fragments of a computation. The heaps we develop here allow for such a representation, and easily generate AES. In this paper, we develop a partial-order semantics in terms of heaps, for Orc, an orchestration language used to describe distributed computations over the internet. We briefly show how Orc, and this new semantics, are used for QoS studies of wide area orchestrations.

1 Introduction

Orchestrating Web services consists of a combination of different activities. A primary concern is to ensure that the expected functionality is indeed correctly implemented. This requires semantic studies for the formalisms used in specifying the functional aspect of Web service orchestrations. Examples of such studies include the translation of the industrial standard BPEL into WorkFlow nets [15] (a special subclass of Petri nets) or the pi-calculus [14], from which analysis techniques and tools for BPEL [13,2] were developed.

Another important, yet much less addressed task consists in ensuring that the Web service orchestration offers the due Quality of Service (QoS). QoS parameters are not firmly established, but they typically include response time (latency), availability, maximum allowed query rate (throughput), and security. The Web Service Level Agreement (WSLA) framework [11] is a standard proposed by IBM for QoS parameters in Web Services. When applied to the management of OEM/supplier cooperations, orchestrations must make precise the duties and

* This work was partially funded by the ANR national research program DOTS (ANR-06-SETI-003), DocFlow (ANR-06-MDCA-005) and the project CREATE ActivDoc.

responsibilities of the different actors in such chains, via contracts [5]. Having contracts with each subcontractor, the orchestration can establish the overall contract with its customers. This process is called contract composition.

We believe there is a need for semantic studies underpinning the design of Web services orchestrations in all its aspects: functional, QoS, and contracts, including contract composition. Developing such a holistic approach can become quickly cumbersome if rich formalisms for describing Web services orchestrations are considered, such as, e.g., BPEL. The functional semantics of BPEL is in itself complex, due to the large number of features offered. Extending such semantics to encompass QoS aspects can be cumbersome. Orc [12] has been recently proposed as a small and elegant language for wide area computing and Web services orchestrations. While keeping small, it offers the main features required by wide area computing, namely: service call, parallel and sequential composition, preemption, and recursion. Orc has been successfully used to model typical workflow patterns defined by Van der Aalst et al [18].

This paper proposes the foundations for an Orc based design of Web services orchestrations, including both functional and QoS aspects, and supporting contract composition. An interleaving semantics, both operational and denotational, was proposed for Orc in [12]. To prepare for a combined functional/QoS use, we propose in this paper a partial order semantics that keeps track of causalities and concurrency. This allows us to address all the aspects of QoS where causality and concurrency relating the different site calls matters. For example, if an orchestration causally depends on a given site call, failure of this site to deliver proper service causes failure of the orchestration. Another example is that of latency: causal dependencies and concurrency between site calls and other events are reflected into the dates of completion of these different events. Companion paper [16] details the use of this semantics for QoS studies and contract composition, and describes the resulting TOrQuE tool (Tool for Orchestration QuaIity of Service Evaluation).

The paper is organized as follows. Section 2 briefly introduces Orc and its operational semantics. Asymmetric event structures and heap semantics of Orc are described in Section 3 where its use in QoS studies is sketched. Related work is given in Section 4.

2 Orc Overview

An Orc program consists of a set of definitions and a goal expression which is to be evaluated. Orc assumes that basic services, like sequential computation and data manipulation, are implemented by primitive sites. Orc provides constructs to orchestrate the concurrent invocation of sites.

The syntax of Orc is given in the upper portion of figure 1. Orc defines three basic operators. For Orc expressions f, g, “f \parallel g” executes f and g in parallel. “f \gg x > g” evaluates f first and for every value returned by f, a new instance of g is launched with variable x assigned to this return value. “f where x : \in g” executes f and g in parallel. When g returns its first value, x is assigned to this