Science of Computer Programming 80 (2014) 3–24

Contents lists available at ScienceDirect

Science of Computer Programming

journal homepage: www.elsevier.com/locate/scico

A compositional model to reason about end-to-end QoS in Stochastic Reo connectors

Young-Joo Moon [a,*], Alexandra Silva [d,b,e], Christian Krause [c], Farhad Arbab [b]

[a] *INRIA, Bordeaux, France*
[b] *Centrum Wiskunde & Informatica (CWI), Amsterdam, The Netherlands*
[c] *Hasso Plattner Institute (HPI), Potsdam, Germany*
[d] *Radboud University Nijmegen, Nijmegen, The Netherlands*
[e] *HASLab/INESC TEC, Universidade do Minho, Braga, Portugal*

ARTICLE INFO

Article history:
Received 22 November 2010
Received in revised form 21 November 2011
Accepted 23 November 2011
Available online 1 December 2011

Keywords:
Coordination language
Reo
Continuous-ime Markov Chains
Quality of service
Compositional semantic model

ABSTRACT

In this paper, we present a compositional semantics for the channel-based coordination language Reo that enables the analysis of quality of service (QoS) properties of service compositions. For this purpose, we annotate Reo channels with stochastic delay rates and explicitly model data-arrival rates at the boundary of a connector, to capture its interaction with the services that comprise its environment. We propose Stochastic Reo Automata as an extension of Reo automata, in order to compositionally derive a QoS-aware semantics for Reo. We further present a translation of Stochastic Reo Automata to Continuous-Time Markov Chains (CTMCs). This translation enables us to use third-party CTMC verification tools to do an end-to-end performance analysis of service compositions. In addition, we discuss to what extent Interactive Markov Chains (IMCs) can serve as an alternative semantic model for Stochastic Reo. We show that the semantics of Stochastic Reo cannot be specified compositionally using the product operator provided by IMCs.

1. Introduction

In service-oriented computing (SOC), complex distributed applications are built by composing existing – often third-party – services using additional coordination mechanisms, such as workflow engines, component connectors, or tailor-made glue code. Due to the high degree of heterogeneity and the fact that the owner of the application is not necessarily the owner of its building blocks, issues involving quality of service (QoS) properties become increasingly entangled. Even if the QoS properties of every individual service and connector are known, it is far from trivial to determine and reason about the end-to-end QoS of a composed system in its application context. Yet, the end-to-end QoS of a composed service is often as important as its functional properties in determining its viability in its market.

Reo [1], a channel-based coordination language, supports the composition of services, and typically, its semantics is given in terms of Constraint Automata (CA) [2]. However, CA do not account for the QoS properties and cannot capture the context-dependency [2] of Reo connectors. To capture context-dependency, Reo Automata were introduced in [3]. However, they also provide no means for modeling QoS properties. On the other hand, Quantitative Intentional Automata (QIA) were proposed in [4] to account for the end-to-end QoS properties of Reo connectors. Unfortunately, no formal results are readily available regarding the compositionality of QIA. Thus, in order to overcome the shortcomings of CA and QIA, mentioned above, the design of a new compositional semantic model for Reo connectors was required.

[*] Corresponding author.
E-mail addresses: young-joo.moon@inria.fr (Y.-J. Moon), alexandra@cs.ru.nl (A. Silva), christian.krause@hpi.uni-potsdam.de (C. Krause), farhad@cwi.nl (F. Arbab).

0167-6423/$ – see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.scico.2011.11.007

For this purpose, in [5], we suggested *Stochastic Reo Automata* as a compositional semantic model for reasoning about the end-to-end QoS properties, as well as handling the context-dependency of Reo connectors. We showed that the compositionality results of Reo Automata extend to Stochastic Reo Automata. We also presented a translation of Stochastic Reo Automata to Continuous-Time Markov Chains (CTMCs). This enabled the use of third-party tools for stochastic analysis. Therefore, [5] shows a compositional approach for constructing Markov Chain (MC) models of complex composite systems, using Stochastic Reo Automata as an intermediate model. Stochastic Reo Automata provide a compositional framework wherein the corresponding CTMC model of a connector can be derived. This approach, thus, enabled us to model the QoS properties of system behavior, where our translation derives a CTMC model for complex systems for subsequent analysis by other tools. This paper is the extended version of [5] together with the contribution mentioned above. In this paper, we provide more examples for Stochastic Reo, its semantic model, and the translation method. We show the proof of the compositionality of Stochastic Reo Automata. In addition, we discuss to what extend Interactive Markov Chains (IMCs) can serve as an alternative semantic model for Stochastic Reo.

2. Overview of Reo

Reo is a channel-based coordination model wherein so-called *connectors* are used to coordinate (i.e., control the communication among) components or services exogenously (from outside of those components and services). In Reo, complex connectors are compositionally built out of basic channels. Channels are atomic connectors with exactly two ends, which can be either *source* or *sink* ends. Source ends accept data into, and sink ends dispense data out of their respective channels. Reo allows channels to be undirected, i.e., to have respectively two source or two sink ends.

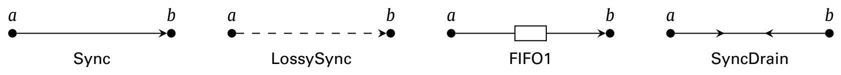

Fig. 1. Some basic Reo channels.

Fig. 1 shows the graphical representations of some basic channel types. The Sync channel is a directed, unbuffered channel that synchronously reads data items from its source end and writes them to its sink end. The LossySync channel behaves similarly, except that it does not block if the party at the sink end is not ready to receive data. Instead, it just loses the data item. FIFO1 is an asynchronous channel with a buffer of size one. The SyncDrain channel differs from the other channels in that it has two source ends (and no sink end). If there is data available at both ends, this channel consumes (and loses) both data items synchronously.

Channels can be joined together using nodes. A node can have one of three types: source, sink or mixed node, depending on whether all ends that coincide on the node are source ends, sink ends or a combination of both. Source and sink nodes, called *boundary nodes*, form the boundary of a connector, allowing interaction with its environment. Source nodes act as synchronous replicators, and sink nodes as mergers. A mixed node combines both behaviors by atomically consuming a data item from one sink end and replicating it to all of its source ends.

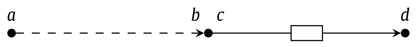

Fig. 2. Example connector: LossyFIFO1.

An example connector is depicted in Fig. 2. It reads a data item from a, buffers it in a FIFO1 and writes it to d. The connector loses data items from a if and only if the FIFO1 buffer is already full. This construct, therefore, behaves as a connector called (overflow) LossyFIFO1.

2.1. Semantics: Reo Automata

In this section, we recall Reo Automata [3], an automata model that provides a compositional operational semantics for Reo connectors. Intuitively, a Reo Automaton is a non-deterministic automaton whose transitions have labels of the form $g|f$, where f a set of nodes that fire synchronously, and g is a *guard* (boolean condition) that represents the presence or the absence of I/O requests at nodes, i.e., the pending status of the nodes. A transition can be taken only when its guard g is true.

We recall some facts about Boolean algebras. Let $\Sigma = \{\sigma_1, \ldots, \sigma_k\}$ be a set of symbols that denote names of connector ports, $\overline{\sigma}$ be the negation of σ, and $\mathcal{B}_\Sigma$ be the free Boolean algebra generated by the following grammar:

$$g ::= \sigma \in \Sigma \mid \top \mid \bot \mid g \vee g \mid g \wedge g \mid \overline{g}$$

We refer to the elements of the above grammar as *guards* and in its representation we frequently omit $\wedge$ and write $g_1 g_2$ instead of $g_1 \wedge g_2$. Given two guards $g_1, g_2 \in \mathcal{B}_\Sigma$, we define a natural order $\leq$ as $g_1 \leq g_2 \iff g_1 \wedge g_2 = g_1$. The intended interpretation of $\leq$ is logical implication: g_1 implies g_2. An *atom* of $\mathcal{B}_\Sigma$ is a guard $a_1 \ldots a_k$ such that $a_i \in \Sigma \cup \overline{\Sigma}$ with $\overline{\Sigma} = \{\overline{\sigma}_i \mid \sigma_i \in \Sigma\}$, $1 \leq i \leq k$. We can think of an atom as a truth assignment. We denote atoms by Greek letters $\alpha, \beta, \ldots$