Journal of Visual Languages and Computing 24 (2013) 153–168

Contents lists available at SciVerse ScienceDirect

Journal of Visual Languages and Computing

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

Compositional and behavior-preserving reconfiguration of component connectors in Reo

Christian Krause [a,1,*], Holger Giese [a], Erik de Vink [b]

[a] Hasso Plattner Institute for Software Systems Engineering, Prof.-Dr.-Helmert-Str. 2–3, D-14482 Potsdam, Germany
[b] Department of Mathematics & Computer Science, Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, The Netherlands

ARTICLE INFO

This paper has been recommended for acceptance by Shi Kho Chang
Available online 5 November 2012

Keywords:
Coordination languages
Reconfiguration
Behavior-preservation
Semantics
Category theory

ABSTRACT

It is generally accepted that building software out of loosely coupled components, such as in service-oriented systems or mobile networks, yields applications that are more robust against changes and failure of single components than monolithic systems. In order to accommodate for changes in the environment or in the requirements, and anticipate to a component failure, applications are often dynamically adapted by means of a reconfiguration. In this paper, we target the visual channel-based coordination language Reo and introduce a combined structural and behavioral model for graph-based component connectors in Reo. Exploiting concepts from category theory, we model reconfigurations of connectors as transformations of the underlying connector graphs. We show that our connector model has a compositional semantics and lift structural reconfigurations to the semantical level. As a concrete application of our framework, we introduce a notion of behavior-preserving reconfiguration for Reo and provide a sufficient condition to ensure behavior-preservation statically.

1. Introduction

A common approach in software engineering is based on the idea of dividing a system into two orthogonal aspects: (i) the computation performed by a set of (black-boxed) components or services, and (ii) their coordination using specific 'glue code'. Engineering systems using this principle has the advantage that there is clear separation between the encapsulated functional behavior implemented in the components on the one hand, and the description of their allowed interactions in the form of communication protocols on the other.

In order to be able to reason about such systems or to (semi-) automatically derive implementations of them, models with precise semantics are vital. While the specification of the components or services is usually based on behavioral models, e.g. automata or process algebraic descriptions, the composition and the interaction of these functional building blocks is often described using graphical models, e.g. various kinds of Petri nets or structural component connector models.

In this paper, we specifically target component connectors in the channel-based coordination language Reo [1]. In Reo, the coordination of components and services is realized using arbitrarily complex connectors which are represented by graphs that consist of a set of communication channels connected by nodes. While the structure of component connectors is graph-based, their semantics can be given in terms of automata, e.g. *constraint automata* [2].

In practice, however, the complexity of the modeled systems and changes in their requirements or protocols demand to adapt them at runtime. Adjusting a component connector to accommodate for changes in the protocol or to reflect new goals, can be achieved by means of a

* Corresponding author. Tel.: +49 3315509525.
E-mail address: christian.krause@hpi.uni-potsdam.de (C. Krause).
[1] Supported by a research grant of the Hasso Plattner Institute for Software Systems Engineering.

1045-926X/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.jvlc.2012.09.002

reconfiguration. Often, reconfiguration not only involves the changing of the runtime parameters of single channels or components, but also requires structural modifications at the level of the component connectors.

In a promising line of research, the theory of graph transformation is used to model reconfigurations at the structural level of systems, e.g. reconfigurations of component connectors [3], Petri nets [5], mobile networks [6] and software architectures [7]. In all these areas, graphs are used because they provide a both natural and formal representation of the structure of the systems in terms of their topology. The idea of graph transformation is to rewrite substructures using matched transformation rules. Such rules provide a syntax to specify structural preconditions in the form of local contexts of system elements, and moreover allow the engineer to define and execute complex reconfigurations as atomic structural modifications.

When using graphs to model the structure of systems and graph transformation to realize their reconfiguration, one of the major challenges is to predict the impact of a structural modification on the behavioral semantics of the systems at hand. For instance, when dynamically adding new channels or nodes to a component connector in Reo, it is important to understand the effect on its execution semantics. In such situations, it is often crucial to ensure that certain behavioral properties are preserved by the reconfiguration.

1.1. Approach and contributions

In this paper, we consider graph transformation-based reconfiguration of component connectors in Reo, as e.g. employed already in [3,4]. In order to reason about the impact of a structural reconfiguration of a Reo connector on its semantics, we present a combined structural and behavioral model for Reo, called *distributed constraint automata with state memory* (DCASM).

Our model permits to specify the topology of a connector together with the semantics of its constituent channels and components and has moreover enough expressive power to derive implementations of Reo that can be used in practice. Specifically, it permits to specify data constraints over infinite data domains by symbolic representations using memory cells as introduced in [8], and moreover carries relevant structural information, i.e., the topology information of the component connectors.

The framework presented in this paper builds on concepts from category theory which facilitates an abstract and elegant way of defining operations and reasoning about properties. Specifically, we define the composition of connector models in our framework using universal properties and phrase the semantics in terms of a compositional functor, which derives behavioral automata models from graphical connector models.

We facilitate the theory of algebraic graph transformation [9] to model reconfigurations as structural transformations of connector graphs as employed in [3,4]. Due to the compositionality of the semantics functor, we are able to lift a structural reconfiguration of a connector graph to the semantical level, i.e., to its automata semantics. Using this approach, we are able to investigate the impact of a structural change in a connector on its overall behavior. As a concrete application of our framework, we introduce a notion of a behavior-preserving reconfiguration and give a sufficient condition to ensure behavior-preservation statically.

This paper extends the results presented in [10] as follows. We generalize the basic model of distributed port automata in [10] to the more expressive model of distributed constraint automata with state memory. Thereby, data constraints for channels (even over infinite data domains) can be modeled in our framework. We transfer the compositionality result for distributed port automata in [10] to the level of distributed constraint automata with state memory in this paper. Furthermore, we show explicitly how graph transformation can be employed to model structural connector reconfigurations and formally define the induced reconfiguration at the semantical level. To the best of our knowledge, none of the existing models for Reo provides a concept for defining and checking behavior-preserving reconfiguration, as introduced in this paper.

1.2. Overview

In Section 2, we recall relevant notions from category theory. We assume familiarity with the fundamental concepts, such as categories, functors and (co)limits. In Section 3, we discuss the modeling concepts for building component connectors in Reo. Specifically, we recall the notions of channels and nodes and explain how they can be composed to construct complex connector graph models. In Section 4, we define the formal semantics of Reo in terms of constraint automata with state memory (CASM) and introduce a notion of simulation to relate two automata. In Section 5, we present a categorical view of the CASM model where we consider automata as objects and simulations as morphisms between objects. Moreover, we define the parallel composition for CASM in categorical terms, i.e., using a universal property. In Section 6, we recall the basic concepts of distributed graph transformation, which are essential in our categorical framework to model the topology of connectors. In Section 7, we combine the categorical models of Sections 5 and 6 to obtain an integrated structural and behavioral model of component connectors, called *distributed constraint automata with state memory* (DCASM). This model builds on concepts from category theory and constitutes the core of our framework. As one of our central results, we show that the semantics of connectors in this model is compositional, i.e., that a structural gluing of connector graphs corresponds to a parallel composition in the automata semantics. In Section 8, we discuss how the theory of graph transformation can be applied in our framework to model the reconfiguration of component connectors. Reconfigurations are specified at the structural level of component connectors, specifically using graph transformation rules that modify the structure of connectors. The compositional semantics of our DCASM model allows us further to formally define the impact of a structural reconfiguration of a connector on its automata semantics. We exploit this correspondence to define a notion of behavior-preserving reconfiguration. We discuss