ARTICLE IN PRESS
JLAMP:119

Journal of Logical and Algebraic Methods in Programming ••• (••••) •••–•••

Contents lists available at ScienceDirect

Journal of Logical and Algebraic Methods in Programming

www.elsevier.com/locate/jlamp

Configuration logics: Modeling architecture styles

Anastasia Mavridou, Eduard Baranov, Simon Bliudze *, Joseph Sifakis

École polytechnique fédérale de Lausanne, Rigorous System Design Laboratory, Station 14, CH-1015, Lausanne, Switzerland

A R T I C L E   I N F O

Article history:
Received 31 October 2015
Received in revised form 12 May 2016
Accepted 13 May 2016
Available online xxxx

Keywords:
Architecture styles
Coordination
Configuration logics
Component interaction
BIP

A B S T R A C T

We study a framework for the specification of architecture styles as families of architectures involving a common set of types of components and coordination mechanisms. The framework combines two logics: 1) interaction logics for the specification of architectures as generic coordination schemes involving a configuration of interactions between typed components; and 2) configuration logics for the specification of architecture styles as sets of interaction configurations. The presented results build on previous work on architecture modeling in BIP. We show how propositional interaction logic can be extended into a corresponding configuration logic by adding new operators on sets of interaction configurations. In addition to the usual set-theoretic operators, configuration logic is equipped with a coalescing operator $+$ to express combination of configuration sets. We provide a complete axiomatization of propositional configuration logic as well as decision procedures for checking that an architecture satisfies given logical specifications. To allow genericity of specifications, we study first-order and second-order extensions of the propositional configuration logic. First-order logic formulas involve quantification over component variables. Second-order logic formulas involve additional quantification over sets of components. We provide several examples illustrating the application of the results to the characterization of various architecture styles. We also provide an experimental evaluation using the Maude rewriting system to implement the decision procedure for the propositional flavor of the logic.

1. Introduction

Architectures are common means for organizing coordination between components in order to build complex systems and to make them manageable. They depict generic coordination principles between components and embody design rules that can be understood by all. Architectures allow thinking on a higher plane and avoiding low-level mistakes. They are a means for ensuring global coordination properties between components and thus, achieving correctness by construction [1].

Using architectures largely accounts for our ability to master complexity and develop systems cost-effectively. System developers extensively use reference architectures ensuring both functional and non-functional properties, e.g. fault-tolerant, time-triggered, adaptive, security architectures.

Many languages have been proposed for architecture description such as architecture description languages, e.g. [2], coordination languages, e.g. [3] and configuration languages [4]. All these works rely on the distinction between behavior of individual components and their coordination in the overall system organization. Informally architectures are characterized

* Corresponding author.
E-mail addresses: anastasia.mavridou@epfl.ch (A. Mavridou), eduard.baranov@epfl.ch (E. Baranov), simon.bliudze@epfl.ch (S. Bliudze), joseph.sifakis@epfl.ch (J. Sifakis).

http://dx.doi.org/10.1016/j.jlamp.2016.05.002
2352-2208/© 2016 Published by Elsevier Inc.

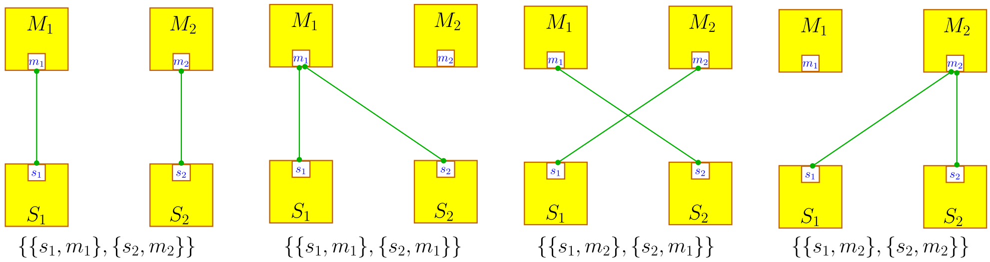

Fig. 1. Master/Slave architectures.

by the structure of the interactions between a set of typed components. The structure is usually specified as a relation, e.g. connectors between component ports.

The field of software architecture remains relatively immature [5]. A lot of foundational issues remain open. One is the distinction between architectures and their properties. Architecture styles characterize not a single architecture but a family of architectures sharing common characteristics such as the type of the involved components and the topology induced by their coordination structure. Simple examples of architecture styles are Pipeline, Ring, Master/Slave, Pipe and Filter. For instance, Master/Slave architectures integrate two types of components, masters and slaves such that each slave can interact only with one master. Fig. 1 depicts four Master/Slave architectures involving two master components M_1, M_2 and two slave components S_1, S_2. Their communication ports are, respectively, m_1, m_2 and s_1, s_2. The architectures correspond to interaction configurations: $\big\{\{s_1, m_1\}, \{s_2, m_2\}\big\}$, $\big\{\{s_1, m_1\}, \{s_2, m_1\}\big\}$, $\big\{\{s_1, m_2\}, \{s_2, m_1\}\big\}$ and $\big\{\{s_1, m_2\}, \{s_2, m_2\}\big\}$. The set $\{s_i, m_j\}$ denotes an interaction between ports s_i and m_j. A configuration is a non-empty set of interactions. The Master/Slave architecture style characterizes all the Master/Slave architectures for arbitrary numbers of masters and slaves.

The paper studies the relation between architectures and architecture styles. This relation is similar to the relation between programs and their specifications. As program specifications can be expressed by using logics, e.g. temporal logics, architecture styles can be specified by configuration logics characterizing classes of architectures.

First, we propose a propositional configuration logic (PCL) whose formulas represent, for a given set of components, the allowed configuration sets. Then, we introduce first-order and second-order logics as extensions of the propositional logic. These allow genericity of description as they are defined for types of components.

The proposed formalism is declarative and has some similarities with languages used for a feature-oriented analysis of architectures, such as OCL [6]. It differs from formalisms used to describe the possible configurations of a dynamic architecture by using graph grammars [7,8].

The meaning of a configuration logic formula is a configuration set. A configuration on a set of components represents a particular architecture. Thus, configuration logic formulas describe architecture sets. The definition of configuration logics requires considering three hierarchically structured semantic domains:

The lattice of interactions. An interaction a is a non-empty subset of P, the set of ports of the integrated components. Its execution implies the atomic synchronization of all component actions (at most one action per component) associated with the ports of a.

The lattice of configurations. Configurations are non-empty sets of interactions characterizing architectures.

The lattice of configuration sets. Sets of configurations are properties described by the configuration logic.

We aim at describing systems of interacting components: in each configuration there must be at least one interaction and each of the interactions should involve at least one component. Therefore, we only consider non-empty interactions and configurations.

Fig. 2 shows the three lattices for $P = \{p, q\}$. For the lattice of configuration sets, we show only how it is generated.

This work consistently extends results on modeling architectures by using propositional interaction logic [9–11], which are Boolean algebras on the set of ports P of the composed components. Their semantics is defined via a satisfaction relation $\models_i$ between interactions and formulas. Each interaction logic formula ϕ represents exactly the set of interactions corresponding to Boolean valuations of P satisfying ϕ.

Configuration logic is a powerset extension of interaction logic. Its formulas are generated from the formulas of the propositional interaction logic by using the operators union, intersection and complementation, as well as a *coalescing operator* $+$. To avoid ambiguity, we refer to the formulas of the configuration logic that syntactically are also formulas of the interaction logics as *interaction formulas*. The semantics of the configuration logic is defined via a satisfaction relation $\models$ between configurations $\gamma = \{a_1, \ldots, a_n\}$ and formulas. An interaction formula f represents any configuration consisting of interactions satisfying it; that is $\gamma \models f$ if, for all $a \in \gamma$, $a \models_i f$. For set-theoretic operators we take the standard meaning. The meaning of formulas of the form $f_1 + f_2$ is all configurations γ that can be decomposed into γ_1 and γ_2 ($\gamma = \gamma_1 \cup \gamma_2$) satisfying, respectively, f_1 and f_2. The formula $f_1 + f_2$ represents configurations obtained as the union of configurations of f_1 with configurations of f_2.

Please cite this article in press as: A. Mavridou et al., Configuration logics: Modeling architecture styles, J. Log. Algebraic Methods Program. (2016), http://dx.doi.org/10.1016/j.jlamp.2016.05.002