Science of Computer Programming ••• (••••) •••–•••

Contents lists available at ScienceDirect

Science of Computer Programming

www.elsevier.com/locate/scico

Simulation-based matching of cloud applications

Filippo Bonchi[b], Antonio Brogi[a], Andrea Canciani[a], Jacopo Soldani[a,*]

[a] Dept. of Computer Science, University of Pisa, Italy
[b] Univ. Lyon, CNRS, ENS de Lyon, UCB Lyon 1, LIP, France

ARTICLE INFO

Article history:
Received 2 January 2017
Received in revised form 5 June 2017
Accepted 6 June 2017
Available online xxxx

Keywords:
Service matching
TOSCA
Reuse
Simulation
Coinduction

ABSTRACT

OASIS TOSCA aims at solving the problem of managing complex applications across heterogeneous clouds by providing a standard, vendor-agnostic language to describe them. TOSCA permits defining a cloud application as an orchestration of typed components, which can be instantiated by matching other TOSCA applications.

In this paper we first present two types of behaviour-aware matching of applications (*exact* and *plug-in*) both based on a notion of simulation. We then extend the notion of plug-in matching by relaxing the notion of simulation to permit matching an operation with a sequence of operations. We also present a coinductive procedure to compute such relaxed simulation, and we formally prove the termination, soundness, and completeness of such procedure.

1. Introduction

Cloud computing has revolutionised IT by permitting to run on-demand distributed applications at a fraction of the cost which was necessary just a few years ago [2]. However, current cloud technologies suffer of a lack of standardization, with different providers offering similar resources in a different manner. How to flexibly manage complex applications across heterogeneous clouds is thus one of the main research problems in the cloud environment [10].

To tackle this issue, OASIS released the *Topology and Orchestration Specification for Cloud Applications* (TOSCA [24]), a standard to describe complex cloud applications, and to support the automation of their management. An application is specified in TOSCA by instantiating component types (possibly by reusing matching applications [13]), by connecting a component's requirements to the capabilities of other components, and by orchestrating the operations of the application components into plans defining the management of the whole application.

Unfortunately, the current specification of TOSCA [24] does not permit to describe the *behaviour* of applications' management operations. More precisely, there is no way to describe the order in which the operations of a component must be invoked, nor how those operations depend on the requirements or affect the capabilities of that component. As a consequence, existing matchmaking and adaptation techniques (e.g., [12,28]) are behaviour-unaware, and this may cause problems when substituting a component by reusing a matched application.

In this paper we first recall how TOSCA can be extended to specify the behaviour of management operations by generalising the model we proposed in [7]. Namely, the management protocols of a TOSCA component can be described by means of finite state machines whose states and transitions are associated with conditions on the requirements and capabilities of such component. The objective of those conditions is essentially to define the consistency of the states of a component, and to constrain the executability of its management operations to the satisfaction of its requirements.

[*] Corresponding author.
E-mail addresses: filippo.bonchi@ens-lyon.fr (F. Bonchi), brogi@di.unipi.it (A. Brogi), canciani@di.unipi.it (A. Canciani), soldani@di.unipi.it (J. Soldani).

http://dx.doi.org/10.1016/j.scico.2017.06.001
0167-6423/© 2017 Elsevier B.V. All rights reserved.

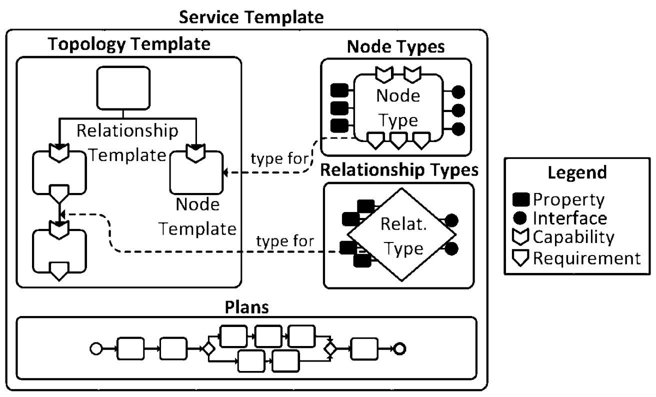

Fig. 1. A TOSCA service template.

We then constrain the existing notions of (syntactic) *exact* and *plug-in* matching in TOSCA [12] to take into account behaviour information in management protocols. More precisely, we define when a desired management protocol can be "simulated" [27] by another management protocol, and we exploit such notion of simulation to constrain exact and plug-in matchings.

We then relax the notion of simulation into that of f-simulation, where f is a function associating each transition in the desired management protocol with a sequence of transitions in the available protocol. This permits to further relax the notion of plug-in matching, so to match operations of a desired component with sequences of operations of an available application. We also describe how flexibly plug-in matched applications can be suitably adapted so to be employed in place of desired components.

Finally, we introduce a coinductive [19] algorithm (called ComputeFs) that permits computing the function f determining an f-simulation between two management protocols. We also coinductively prove that ComputeFs is terminating, sound and complete.

This paper is an extended version of [5], including (i) a more detailed background on TOSCA, (ii) a formalisation of the coinductive algorithm Compute Fs, and (iii) the formal assessment of termination, soundness, and completeness of ComputeFs.

The rest of the paper is organised as follows. Sect. 2 provides the necessary background on TOSCA, while Sect. 3 describes how TOSCA can be extended to specify the behaviour of management operations. Sect. 4.1 recalls the notions of (syntactic) *exact* and *plug-in* matching. Sect. 4.2 introduces the notion of management protocol simulation, and it employs such notion to further constrain the *exact* and *plug-in* matchings. Sect. 4.3 relaxes the notion of simulation into that of f-simulation to further relax *plug-in* matching. Sect. 5 presents the algorithm to compute f-simulations and formally assess its termination, soundness, and completeness. Finally, Sects. 6 and 7 discuss related work and draw some conclusions, respectively.

2. Background: TOSCA

TOSCA [24] is an OASIS standard aimed at enabling the specification of portable cloud applications and the automation of their management. To do so, TOSCA provides a modelling language to describe the structure of a cloud application as a typed topology graph, and its tasks as plans. More precisely, each cloud application is represented as a *service template* (Fig. 1), consisting of a mandatory *topology template* and of optional management *plans*.

The topology template is a typed directed graph describing the structure of the composite cloud application. Its nodes (called *node templates*) model the application components, while its edges (called *relationship templates*) model the relations among those components. Node templates and relationship templates are typed by means of *node types* and *relationship types*, respectively. A node type defines (i) the observable properties of an application component, (ii) the possible states of its instances, (iii) its requirements, (iv) the capabilities it offers to satisfy other components' requirements, and (v) its management operations. Relationship types describe the properties of relationships occurring among components. Requirements, capabilities, properties and operations externally exposed by a service template can be described in its *boundary definitions*.

Plans instead permit describing the management aspects of a service template. Each plan is a workflow orchestrating the management operations offered by the application components to address (part of) the management of the whole cloud application.[1]

Example 2.1. We hereby exemplify how a Moodle application can be specified as a TOSCA service template. MoodleApplication (Fig. 2) is composed by six components, each represented by a node template in its topology. Moodle is the front-end of the application and Database is its back-end. The other four components (viz., a Server, a MySQL runtime environment,

[1] A more detailed, self-contained introduction to TOSCA can be found in [13].