Available online at www.sciencedirect.com

SCIENCE DIRECT®

The Journal of Logic and Algebraic Programming 70 (2007) 15–33

THE JOURNAL OF LOGIC AND ALGEBRAIC PROGRAMMING

www.elsevier.com/locate/jlap

Structural and behavioural compatibility of graphical service specifications [☆], [☆☆]

R. Heckel[a,*], A. Cherchago[b]

[a] Department of Computer Science, University of Leicester, Leicester LE1 7RH, United Kingdom
[b] International Graduate School "Dynamic Intelligent Systems", University of Paderborn, Warburger Str. 100, Paderborn 33098, Germany

Abstract

The ability of applications to dynamically discover required services is a key motivation for Web Services. However, this aspect is not entirely supported by current Web Services standards. It is our objective to develop a formal approach, allowing the automation of the discovery process. The approach is based on the matching of interface specifications of the required and provided services.

In the present paper, we establish an integral notion of structural and behavioural compatibility of service specifications. While structural information is represented by operation declarations, behavioural descriptions are provided by contracts expressed as graph transformation rules with positive and negative application conditions. The integration of structural and behavioural descriptions is facilitated by *typed and parameterised* graph transformation systems, augmenting the rule-based description of behaviour by a type graph and operation declarations representing the structural aspect.

The matching relation taking into account this combination is called *parameterised substitution morphism*. We show that substitution morphisms satisfy the semantic requirement inherent in its name: the substitutability of abstract operations by (calls to) concrete ones.

Keywords: Web services; Service specification matching; Graph transformation; Conditional parameterised rules; Substitutability

1. Introduction

The Web Services platform provides the means to adopt the World Wide Web for application integration based on standards for communication, interface description, and discovery of services. The prosperity of this technology strongly depends on the ability of applications to discover useful services and select those that can be safely integrated with existing components. Much work has been done to achieve this aim. The interface of an offered service can be specified in the Web Service Description Language (WSDL). This specification along with some keywords characterising the

☆ Research funded in part by European Community's Human Potential Programme under contract HPRN-CT-2002-00275, [SegraVis].
☆☆ Research funded in part by International Graduate School "Dynamic Intelligent Systems" of University of Paderborn.
* Corresponding author. Tel.: +44 116 252 3406; fax: +44 116 252 3915.
E-mail addresses: reiko@mcs.le.ac.uk (R. Heckel), cherchago@upb.de (A. Cherchago).

1567-8326/$ - see front matter © 2006 Published by Elsevier Inc.
doi:10.1016/j.jlap.2006.05.003

service can be published at a UDDI-registry which serves as a central information broker and supplies this information to potential clients. However, current standards do not support the automation of checking behavioural compatibility of the requestor's requirements with service descriptions.

The academic community has proposed a number of approaches describing the behaviour of services by *contracts* [1], usually expressed by means of pre- and post-conditions in some form of logic (see Section 4 for a discussion). The main obstacle of this idea is its lack of usability by professional software engineers, whose skills in applying logic formalisms are scarce. In [2,3,4] it has been observed that graph transformation rules could provide a more abstract, visual representation of contracts, specifying preconditions and effects of operations by graphical patterns. This has the advantage of blending intuitively into standard modelling techniques like the UML, providing contracts with an operational (rather than purely descriptive) flavour.

However, the use of the graph transformation technique for Web service contracts is hampered by two problems. Firstly, the classical semantics of rules (as formalised, for example by the double-pushout (DPO) approach to graph transformation [5]) assumes that transformations are completely described by rules, i.e., nothing changes beyond what is explicitly required. A contract, however, represents a potentially incomplete specification of an operation, rather than its implementation, and the strict rule semantics does not reflect the loose nature of contracts. Secondly, classical graph transformation rules do not contain information on the signatures of the specified operations, like input and output parameter types needed to ensure type safety in the interaction.

For the first problem, we propose to use the loose semantics of rules based on *graph transitions*. This kind of semantics has been formalised in the double-pullback (DPB) approach [6] which defines graph transitions as a generalisation of DPO transformations, allowing changes that are not required by the rules. In order to increase the expressiveness of our graphical contract language, we consider rules with positive and negative application conditions. Negative conditions are well-known to increase the expressive power of rules [7]. In the classic approaches, positive application conditions can be encoded by extending both the left- and the right-hand side of a rule by the required elements: they become part of the context. This is no longer possible, in the presence of unspecified effects. In fact, the implicit frame condition, that all elements present before the application and not explicitly deleted, are still present afterwards, is no longer true. Thus, an element matched by a positive application condition may disappear, while an element which is shared between the left- and the right-hand side must be preserved.

Since a service may contain several operations, contracts represented by conditional rules are collected in *conditional graph transformation systems* (GTS) providing the *behavioural* specification of the entire service. All conditional rules of GTS are constructed over a conceptual data type model, represented as a type graph.

To tackle the second problem, the *structural* description of a service is given by a *signature* similar to those known from algebraic specifications [8]. A signature comprises a type graph and operation declarations. An *integral* service description is obtained by an amalgamation of the structural and behavioural specifications, i.e., service signature and conditional GTS, and called *conditional parameterised GTS*. Each rule of such system is equipped with the parameters corresponding to a declaration of the specified operation.

In our work a *compatibility* of provided and required services is defined via the compatibility of operations constituting the service interfaces: For all required operations it is necessary to find structurally and behaviourally compatible provided operations. Structural compatibility requires a correspondence between service signatures, which is captured by a *signature morphism*. Behavioural compatibility amounts to relate the required and provided conditional GTSs. We establish semantic requirements underlying the contract resemblance and use them for the construction of an appropriate relation. This relation, formally defined by a *substitution morphism*, allows to match service specifications developed in different type contexts, which is the first main contribution of the paper. We demonstrate that substitution morphisms satisfy the semantic requirements.

In our previous works (cf. [2,3]), structural and behavioural aspects of compatibility have been considered separately, the current presentation introduces the notion of integral compatibility absorbing the two compatibility aspects. The intended correspondence between integral specifications of services is modeled by a *parameterised substitution morphism*.

The rest of the paper is organised as follows: After presenting in the next section the constituents of service specifications and their formalisation in terms of graph transformation, specification matching is treated in Section 3. In Section 4 we discuss related approaches, and in Section 5 we summarise the main achievements and list issues for future work.