Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 112 (2017) 643–652

Procedia Computer Science

www.elsevier.com/locate/procedia

International Conference on Knowledge Based and Intelligent Information and Engineering Systems, KES2017, 6-8 September 2017, Marseille, France

Engineering Polynomial-Time Solutions for Automatic Web Service Composition

Paul Diac[a]

[a] *Faculty of Computer Sicence, Alexandru Ioan Cuza University, General Berthelot, 16, Iasi, 700483, Romania*

Abstract

Web Service Composition (WSC) is the task of creating some new functionality over a repository of independent resources, Web Services in particular. Services are described by their input and output parameters that are matched if they have the same textual name in service definition for the classic, simplified version of WSC. The problem requires finding an ordered list of services such that all input parameters are available starting from the initially user known parameters and revealing all user required parameters. In this paper we propose a proven efficient polynomial-time solution to Automatic WSC combined with a heuristic for shortening the solution length: the number of Web Services in the composition. The algorithm is tested against several benchmarks of tests and is compared with previous solutions that use AI Planning, revealing tremendous improvements. Two benchmarks are well-known in WSC literature but due to lack of high run time variations over their tests, a new benchmark is created with a special designed generator described in the paper. The new tests reveal more meaningful information.

Keywords: automatic web service composition; polynomial time; heuristic; planning; benchmark

1. Introduction

Web Services are present in many modern software architectures and they behave as independent components that generally solve only one precise task. More complex systems can take use of them only in a composed manner, chaining or branching them with respect to the services requirements [1]. Several standardization languages have been implemented to enable this, such as WS-BPEL [2], BPEL for Semantic Web Services (BPEL4SWS) [3], OWL-S [4] and more. Enabling the automation of the process can be a complex task from multiple perspectives, especially in future where the number of publicly available Web Services is expected to have a steady growth. The focus in this paper is on classical Automatic WSC problem where a repository of Web Services and a single user request are known. Each service consists of its input and output parameters and the user request has the same structure: the initially parameters known by the user and the parameters to be found. For the user request we need to find a list of services with the

* Corresponding author. Tel.: +04-075-215-3555;
E-mail address: paul.diac@info.uaic.ro

1877-0509
10.1016/j.procs.2017.08.110

property that each service in the list can be called: its input parameters are known from the previous services or from the user request and all user required parameters are learned. The problem's solution is a satisfying composition. Solution length is taken into account, shorter solutions are desirable if multiple exist.

The paper is organized as follows: section 1 contains this introduction and section 2 describes the formal definition of the problem. Section 3 shortly presents the previously planning based solutions for WSC. Section 4 presents our evaluation plan: existing benchmarks and their limitations together with a new test generator that is more relevant to running time variations. The proposed solution in section 5 describes our polynomial-time algorithm for WSC with the required data structures and enhanced with section 6's heuristics to shorten the solution length. Last section 7 discusses the experimental results with the conclusion and possible future work.

2. Automatic WSC problem definition

Automatic Web Service Composition can be abstracted to the problem described below. The problem is defined using a graph-like structure with complex nodes and the restrictions on the node path that represents the solution in Web Service Composition.

Node. A node n is defined by a pair $\langle I,O\rangle$ where I and O are sets of parameters. I is the input parameter set for the node and O is the output parameter set. We will write them as $n.I$ and $n.O$.

Parameter set. The set of all parameters that appear in all nodes as input or output; also called *the universe*. If R is the set of all nodes then it is $\bigcup_{n\in R} n.I \cup n.O$.

Initial node. A special node *Init* that specifies the initially known parameters. *Init.I* should conventionally be $\emptyset$ and $i.O$ the set of initially know parameters. This way the user request input parameters can be represented as a regular node.

Goal node. A special node *Goal* that defines the parameters that need to be found. *Goal.O* should be $\emptyset$ since it produces no information, just specifies that *Goal.I* is the set of desired parameters. Informally it is the node that needs to be reached.

Parameter matching. Let P be a set of parameters and n a node. We say that the set P matches the node n if $n.I \subseteq P$. We further define $P \oplus n = P \cup n.O$ as the union of $n.O$ and P under the constraint of P matching n.

Chained matching. If P is a set of parameters and $\langle n_1, n_2, ...n_k\rangle$ is an ordered list of nodes, we say that $P \oplus n_1 \oplus n_2 ... \oplus n_k$ is a chain of matching nodes over the set P if:

$$n_i.I \subseteq \left(P \cup \left(\bigcup_{j=1}^{i-1} n_j.O\right)\right), \;\; \forall \;\; i = \overline{1..k}$$

In words, a chain of matching nodes is a list of nodes for which the input of each node is included in the union of the output sets of each previous nodes and the initial set of parameters.

Node Composition problem. Given a set of nodes R and two initial and goal nodes *Init* and *Goal* find a matching list nodes $\langle Init, n_1, n_2, ...n_k, Goal\rangle$ with $n_i \in R$, $\forall\ i = \overline{1..k}$.

Clearly, each node can be interpreted as a Web Service so that Node Composition is equivalent to **WSC**. One Web Service can be executed only if we know all its input parameters. Therefore one valid **WSC** is translated into node chain matching.

Web Service Composition Example. Suppose that as part of a text processing phase we need to replace the predicate of a sentence with a synonym of the verb that constitutes the predicate. The replacement has to be made in the correct conjugation. However, the service that provides synonyms takes as input not a word but a word sense and and there is also a word sense disambiguation service. More precisely, considering the web services:

$$getWordSense \begin{array}{l} in = \{\text{textualWord, sentence}\} \\ out = \{\text{wordSense}\} \end{array} \quad getSynonim \begin{array}{l} in = \{\text{wordSense}\} \\ out = \{\text{word}\} \end{array} \quad getPredicate \begin{array}{l} in = \{\text{sentence}\} \\ out = \{\text{textualWord}\} \end{array}$$

$$getVerbProp \begin{array}{l} in = \{\text{textualWord}\} \\ out = \left\{\begin{array}{c}\text{person, tense,} \\ \text{number, mood}\end{array}\right\} \end{array} \quad conjugateVerb \begin{array}{l} in = \left\{\begin{array}{c}\text{word, person, tense,} \\ \text{number, mood}\end{array}\right\} \\ out = \{\text{conjugatedVerb}\} \end{array}$$