A Hybrid Approach to Web Service Composition Problem in the PlanICS Framework

Artur Niewiadomski¹, Wojciech Penczek¹,², and Jaroslaw Skaruz¹

¹ ICS, Siedlce University, 3-Maja 54, 08-110 Siedlce, Poland
{artur.niewiadomski,jaroslaw.skaruz}@uph.edu.pl
² ICS, Polish Academy of Sciences, Jana Kazimierza 5, 01-248 Warsaw, Poland
penczek@ipipan.waw.pl

Abstract. The paper deals with the concrete planning problem – a stage of the Web Service Composition in the PlanICS framework. A novel (hybrid) planning technique based on a combination of a Genetic Algorithm and a Satisfiability Modulo Theories Solver is introduced. The experimental results of the hybrid algorithm are compared with these obtained using “pure” planning methods.

1 Introduction

Service-Oriented Architecture (SOA) [2] exploits the idea of composing simple functionalities, accessible via well-defined interfaces, in order to satisfy more sophisticated objectives. The problem of finding such a composition is hard and known as the Web Service Composition (WSC) problem [1,2,9].

The system PlanICS [4] is a framework aimed at WSC, which allows for adapting existing real-world services. The main assumption in PlanICS is that all the web services in the domain of interest as well as the objects that are processed by the services, can be strictly classified in a hierarchy of classes, organised in an ontology. Another key idea is to divide the planning into several stages. The first phase deals with classes of services, where each class represents a set of real-world services, while the other phases work in the space of concrete services. The first stage produces an abstract plan composed of service classes [5]. Next, the offers are retrieved by the offer collector (OC) (a module of PlanICS) and used in the concrete planning (CP). As a result of CP a concrete plan is obtained, which is a sequence of offers satisfying predefined optimization criteria. Such an approach enables to reduce dramatically the number of web services to be considered, and inquired for offers.

This paper deals with the concrete planning problem, shown to be NP-hard [7]. Our previous papers employ several techniques to solve it: a genetic algorithm (GA) [10], numeric optimization methods [8], and Satisfiability Modulo Theories (SMT) Solvers [7]. The results of our extensive experiments show that the proposed methods are complementary, but every single one suffers from

* This work has been supported by the National Science Centre under the grant No. 2011/01/B/ST6/01477.
some disadvantages. This observation is the motivation to combining the power of SMT with the potential of GA, which is the main contribution of this paper.

The principal disadvantage of an SMT-based solution is often a long computation time, which is not acceptable in the case of a real-world interactive planning tool. On the other hand, a GA-based approach is relatively fast, but it yields solutions, which are far from optimum and of low probability. Thus, our aim is to exploit the advantages of both methods by combining them into one hybrid algorithm. In the paper we present two new hybrid algorithms and compare their efficiency with the pure SMT- and GA-based planner on several benchmarks.

The rest of the paper is structured as follows. In Section 2 the PlanICS framework is introduced and the Concrete Planning Problem (CPP) is defined. Section 3 presents the main ideas of our hybrid approach as well as some technical solutions. Next, the preliminary experimental results are presented and discussed, followed by conclusions.

2 Concrete Planning Problem

This section introduces the main ideas of the Planics framework and gives all the necessary definitions for defining the concrete planning problem.

An ontology contains a system of classes describing the types of the services as well as the types of the objects they process. A class consists of a unique name and a set of the attributes. By an object we mean an instance of a class. By a state of an object we mean a valuation of its attributes. A set of objects in a certain state is called a world. A key notion of Planics is that of a service. We assume that each service processes a set of objects, possibly changing values of their attributes, and produces a set of new (additional) objects. We say that a