WARP: Efficient Automatic Web Service Composition

Paul Diac
Faculty of Computer Science
Alexandru Ioan Cuza University of Iasi, Romania
Email: paul.diac@info.uaic.ro

***Abstract*—WARP is a new algorithm designed to achieve highest performance in Automatic Web Service Composition (WSC). In its first version, WARP solves the simplest form of WSC, providing very low running times but relative long compositions. Further, a heuristic score is associated to services that promote the most relevant services when multiple are accessible. After this improvement, compositions are short and computed almost as fast as in the first version. Two well-known benchmarks [1], [4] are used to compare WARP with previous planning based approaches [5]. More revealing instances are generated by a proposed test generator.**

***Keywords*-web service composition; orchestration; heuristic;**

I. Introduction

Web Service Composition (WSC) generally refers to the possibilities of using modular, independent and self described resources in a composed manner such that a new specific functionality is exposed. The components are typically Web Services publicly exposed using standard description languages such as WSDL (Web Service Description Language), OWL-S (Web Ontology Language for Services), and others. WSC is a key component in enterprise application integration, being often part of the middleware used for improving connectivity in a variety of distributed systems, such as IoT solutions or B2B frameworks.

II. Problem definition

Automatic WSC focuses on the ways in which a composition can be generated automatically, based on the prior knowledge of a *repository* of services or a Web Service Discovery component. The composition is an ordered subset of available services that can be called in the specified order, which can have a list or a graph-like form.

One particular service can be called at a given time if all his input parameters are known. After the call, all the service output parameters are learned. The initial user request, or the new functionality to be resolved by the composition is a pair of initially known and finally required parameters. All parameter matchings are restricted in the simplest form by the string equality of their names. The output of one service can be input for another if their names coincide in the services definition. In more expressive models, the parameter matchings depend on a degree of similarity of the name or concept, or type matching with respect to a predefined ontology. One of most commons metrics for WSC is the provided composition length. Obviously a smaller number of required Web Services is preferable - if multiple solutions exist.

III. Efficient WSC solution.

In this poster we present *Warp*, a novel solution for WSC addressing the simplified version of the problem, with exact parameter name matching. We develop an algorithm that independent of the solution length is able to compute the composition in **linear time** complexity relative to the problem input size, which is the number of parameters of all services and of the user query. Further, a simple score-based heuristic is added without significantly increasing the complexity and proven to shorten the solution length reaching almost the shortest known compositions length on all tests.

IV. Evaluation.

We first evaluate our solution using two public benchmarks: the ICEBE 2005 [1] and tests generated with the WSBen [4] tool. Since both provide only instances that allow short composition solutions, thus relative short running times for solvers, a new special test generator algorithm was implemented that reveals higher variations.

The algorithm performance is compared with the one of the most popular known alternative solutions to WSC, the AI planning approach. This transforms the WSC instance into a planning domain problem, by associating each service with an *action* in planning. The action expresses *preconditions* and *effects* through a logic predicate *have()* over a set of constants, each representing one WSC parameter. This is presented in [5] as having good results. We use two different planners to solve the instances: GraphPlan [2] and Fast-Forward Planner [3] and compare them with our algorithm.

The performance of our solution is much better than the performance of the planning approach. This is clear enough on the well-known benchmarks but even more on the generated tests. The solution length is only adding at most two extra services on only one of many tests.

V. Conclusion.

The poster describes the test generator, the design of the proposed algorithm and a short analysis of the performance results. Our future work is directed towards adapting the algorithm for more expressive problem definitions that are of interest in the present.

REFERENCES

[1] Web Services Challenge at the IEEE Conference on e-Business Engineering, 2005.

[2] BLUM, A. L., AND FURST, M. L. Fast planning through planning graph analysis. *Artificial intelligence 90*, 1 (1997), 281–300.

[3] HOFFMANN, J. Ff: The fast-forward planning system. *AI magazine 22*, 3 (2001), 57.

[4] OH, S.-C., KIL, H., LEE, D., AND KUMARA, S. R. WSBen: A web services discovery and composition benchmark. In *Web Services, 2006. ICWS'06. International Conference on* (2006), IEEE, pp. 239–248.

[5] ZOU, G., GAN, Y., CHEN, Y., AND ZHANG, B. Dynamic composition of web services using efficient planners in large-scale service repository. *Knowledge-Based Systems 62* (2014), 98–112.

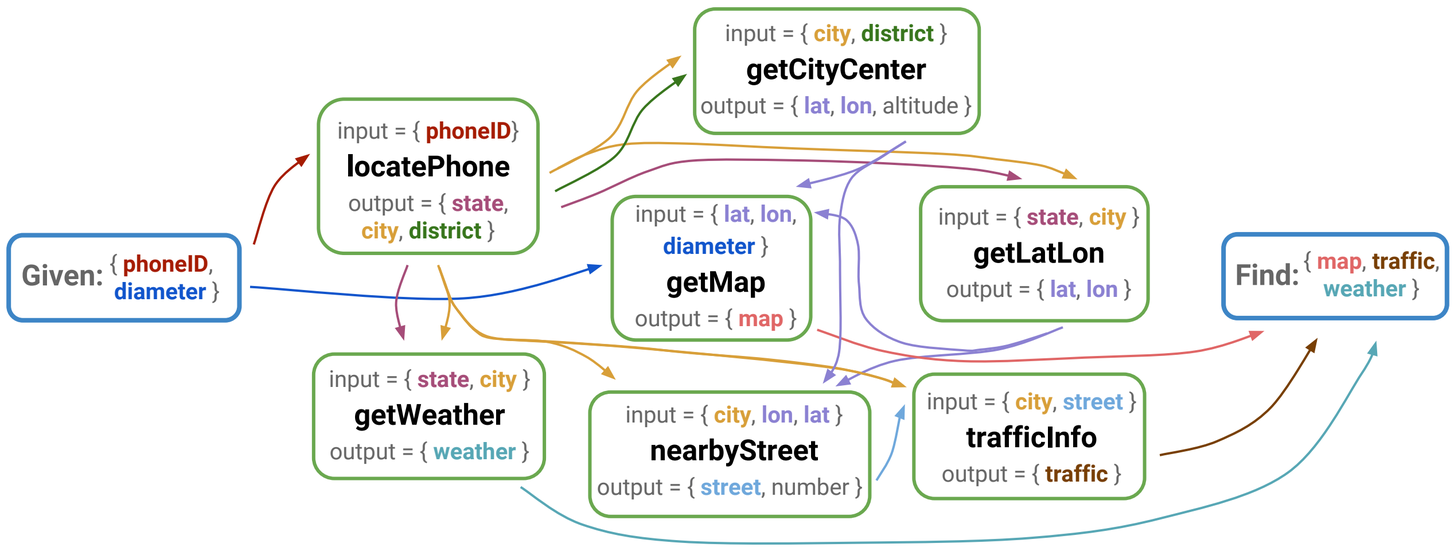
WARP: Efficient Automatic Web Service Composition

Paul Diac • Alexandru Ioan Cuza University of Iasi • paul.diac@info.uaic.ro

Service orchestration • **Heuristic scoring** • **Performance evaluation** • **Benchmark generator**

Web Service Composition (WSC) is a key component in enterprise application integration, being often part of the middleware used for improving connectivity in a variety of distributed systems, such as IoT solutions or B2B frameworks. WARP is designed to achieve high performance in Automatic WSC. In its first version, WARP solves the simplest form of WSC, providing very low running times but relative long compositions. Further, a heuristic score is associated to services that promote the most relevant choices when multiple are available. After this improvement, compositions are short without affecting the efficiency. Two well-known benchmarks, [1] and [4] are used to compare WARP with planning based approaches [5]. More revealing instances are created by the new test generator with special long compositions.

user input — web services repository — user output

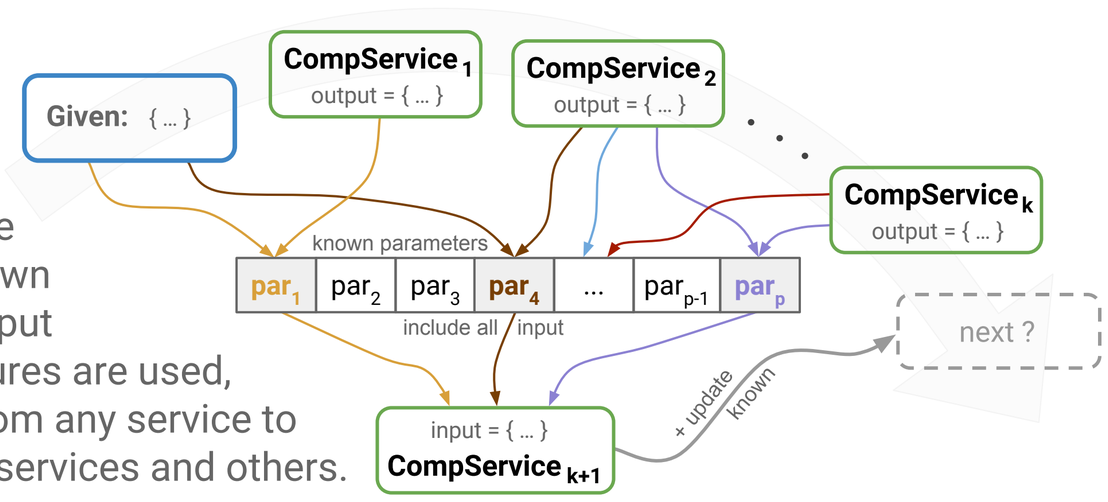


Composition example [5]

Starting with the initially known parameters on the left, **Given : {...}**, the goal on the right, **Find : {...},** has to be reached. There are seven web services. It is possible to find a valid, satisfying composition of a minimum number of six services. Parameters with the same name can be matched between services, from one service output to another service input as indicated by the colored arrows.

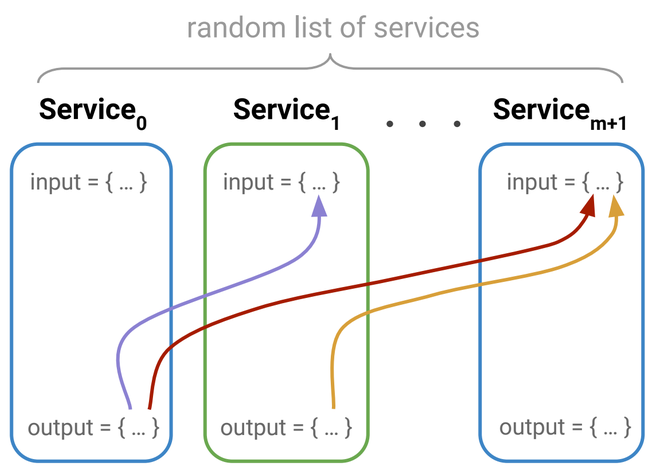
WARP algorithm

The composition is built by adding one service each iteration, starting from the user request input parameters. A service can be chosen next if its input parameters are completely included in the output of all previous services in the composition or the initially known parameters. The service with the highest score is prioritized. Its output parameters are marked accordingly as known. Efficient data structures are used, for example: a **set<>** of all currently known parameters; a **map<,>** from any service to its yet unknown input parameters; a **priority queue<>** of accessible services and others.



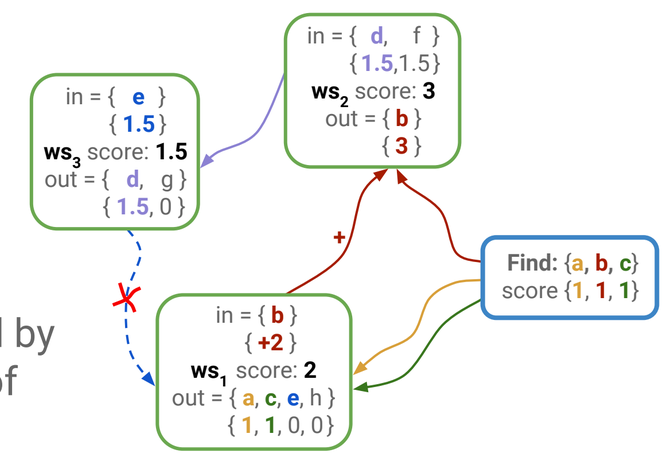
Test generator

Any service in the list has as input a subset of previous services output, implying dependencies. Services **1**...**m** constitute the solution. Service **0** output and **m+1** input constitute the user **Given** and **Find** parameter sets.



Score assignment

Parameters and services are assigned score values proportional to chances of reaching the goal when one service is chosen over another, or a parameter is learned. Loops are avoided by disallowing the repetition of any service in the process.



Results

ICEBE 05 tests[1]

|R|	file	WARP time	WARP length	GraphPlan[2] time	GraphPlan[2] length	Fast-Fwd[3] time	Fast-Fwd[3] length
2656	comp.1-50-16	0.46	2	0.3	2	2.99	2
4156	comp.1-100-4	0.26	4	0.64	4	2.6	4
5356	comp.2-50-32	2.1	7	6.6	7	14.3	7
8356	comp.2-100-32	3.6	7	3.0	7	22.1	7

WARP running times are better than Fast-Fwd and the length is constant.

Test generator

|R|	m	parameters per service	WARP time	WARP len	GraphPlan[2] time	GraphPlan[2] len	Fast-Fwd[3] time	Fast-Fwd[3] len
300	100	15	0.07	50	1.5	51	3.3	50
300	100	40	0.2	95	61	98	43	95
200	150	70	0.3	141	≥ 3hr	?	22min	141
1000	200	50	1.6	130	39.9	133	error	
1000	500	20	0.5	314	≥ 3hr	?	error	
1000	100	20	0.4	86	4.8	87	error	

Running times differ even more on tests where valid compositions are much longer. On larger tests planning either completely fails or runs several minutes while **WARP** can provide shortest known solutions in seconds.

WSBen tests[4]

|R|	WARP time	WARP length	GraphPlan[2] time	GraphPlan[2] length	Fast-Fwd[3] time	Fast-Fwd[3] length
300	0.03	9	0.08	9	0.07	9
1000	0.1	7	0.3	11	1.4	6
5000	0.7	8	2.7	13	4.6	6
10000	1.7	9	3.6	15	error	

On two WSBen tests the composition could be shorter with 1 or 2 services.

good | neutral | bad | very bad

color code

References

[1] Web Services Challenge at the IEEE Conference on e-Business Engineering, 2005.
[2] Blum, A. L., and Furst, M. L. Fast planning through planning graph analysis. Artificial intelligence 90, 1 (1997), 281–300.
[3] Hoffmann, J. Ff: The fast-forward planning system. AI magazine 22, 3 (2001), 57.
[4] Oh, S.-C., Kil, H., Lee, D., and Kumara, S. R. WSBen: A web services discovery and composition benchmark. In Web Services, 2006. ICWS'06. International Conference on (2006), IEEE, pp. 239–248.
[5] Zou, G., Gan, Y., Chen, Y., and Zhang, B. Dynamic composition of web services using efficient planners in large-scale service repository. Knowledge-Based Systems 62 (2014), 98–112.

SYNASC 2017