International Journal of Information Processing, 9(1), 28-37, 2015
ISSN : 0973-8215
IK International Publishing House Pvt. Ltd., New Delhi, India

Usages of Composition Search Tree in Web Service Composition

Lakshmi H N[a], Hrushikesha Mohanty [a]

[a]University of Hyderabad, Hyderabad, Contact: hnlakshmi@gmail.com

The increasing availability of web services within an organization and on the Web demands for efficient search and composition mechanisms to find services satisfying user requirements. Often consumers may be unaware of exact service names that's fixed by service providers. Rather consumers being well aware of their requirements would like to search a service based on their commitments (inputs) and expectations (outputs). Based on this concept we have explored the feasibility of I/O based web service search and composition in our previous work[1]. The classical definition of service composition, *i.e.*, one-to-one and onto mapping between input and output sets of composing services, is extended to give rise to three types of service match: Exact, Super and Partial match. Based on matches of all three types, different kinds of compositions are defined: Exact, Super and Collaborative Composition. Process of composition, being a match between inputs and outputs of services, is hastened by making use of information on service dependency that is made available in repository as an one time preprocessed information obtained from services populating the registry. Adopting three schemes for matching for a desired service outputs, the possibility of having different kinds of compositions is demonstrated in form of a Composition Search Tree. As an extension to our previous work, in this paper, we propose the utility of Composition Search Tree for finding compositions of interest like leanest and the shortest depth compositions.

1. INTRODUCTION

Web Services are self-contained, self-describing, modular applications that can be published, located, and invoked across the Web. As growing number of services are being available, searching the most relevant web service fulfilling the requirements of a user query is indeed challenging.

Various approaches can be used for service search, such as, searching in UDDI, Web and Service portals. The techniques for searching web services can be divided into two categories: discovery and composition. By service composition, we mean making of a new service (that does not exist on its own) from existing services. It can be useful when we are looking for a web service for given inputs and desired outputs and there is no single web service satisfying the request[1].

Most of the existing algorithms [2–8] for service composition construct chains of services based on exact matches of input/output parameters to satisfy a given query. However, the making of a chain fails at a point when inputs of a succeeding (I^S) service does not match exactly with the outputs (O^P) of a preceeding service.

To alleviate this problem, in [1] we propose a Collaborative Composition among such partially matching services for satisfying a desired service outputs, by making match criteria flexible. In addition to exact match we allow partial as well as super match for conditions $O^P \subset I^S$ and $O^P \supset I^S$ respectively. Partial match is of our interest and in [1] we have shown the possibility of successful service composition by collaboration of services that make only partial matches. The process of service composition is visualized as a Composition Search Tree [1] that arranges services in levels showing the way service compositions can be made to meet the user requirements. Our approach[1] results to a scalable implementation for use of RDBMS, a well proven technology.

Here, as an extension to our previous work, we

Input: *CompositionSearchTree*
Output: *SolutionNode* for Shortest Depth Composition
`// Initialization Steps`
CurrentNode = *RootNode* of *CST*
Level = 0
CArr and *NLCArr* are arrays of *ChildNodes*
Insert all *ChildNodes* of *CurrentNode* to *CArr*
while *CArr is not empty* **do**
  Level = *Level* + 1
  foreach *ChildNode in CArr* **do**
    CN=*ChildNode*
    if *CN is an SolutionNode* **then**
      return *[CN]*
    if *CN is an UnsolvableNode* **then**
      Go To 6
    else*CN* is an *UnResolvedNode*
      Insert all *ChildNodes* of *CN* to *NLCArr*
  CArr=*NLCArr*
  CArr=0
return *[NULL]* `// Composition Not Found`

Algorithm 2: Searching Shortest Depth Composition

of the Composition Search Tree. If this *SolutionNode* has the property that it appears at a *Level i* that is equal to *NWS*, then this node will be returned as *SolutionNode* from Line number 13 in Algorithm 1, since this node has the least *NWS* among all *SolutionNodes*.

- **Observation 2:** A *SolutionNode* represents a *LeanestComposition* if and only if there are no other *SolutionNodes* in the Composition Search Tree that has a lesser *NWS* than this *SolutionNode*. **Rationale**: This observation can be reduced from Line numbers 13 and 20 in Algorithm 1. These statements search for the *SolutionNode* with the least *NWS* and hence the algorithm always returns a *SolutionNode* that has the least *NWS*.

5. CONCLUSIONS

This paper is an extension to our previous work in [1]. In [1] the scope of composition is widened defining possibly three modes of service composability: Exact, Partial and Super. Based on composability of all three types and sequencing them differently, *CompositionSearchTree* explores all possible compositions for a given requirement. In the current work, we propose the utility of *CompositionSearchTree* for finding optimal service compositions like *Leanest Composition* and *ShortestDepthComposition*.

The set of web services returned by the algorithms in sections 4 and in [1] implicitly includes the final composition plan when Exact and Super composition or a combination of the two are involved, given by a chain of service calls from the *SolutionNode* till the *RootNode*. However, a composition plan needs to be derived from the set of services whenever the composition includes Collaborative Composition. In the future work we would like to work on an algorithm that generates a composition plan specifying the order of execution for services participitating in a Collaborative composition. Since our system explores all possible compositions for a given requirement, we

would like to include a monitoring component that monitors execution of composition and suggests an alternative composition in case of any service failure.

REFERENCES

1. Lakshmi H N and Mohanty H. RDBMS for Service Repository and Composition, *Fourth International Conference on Advanced Computing (ICoAC)*, pages 13–15, Dec. 2012.
2. S Hashemian, F Mavaddat. A Graph-based Framework for Composition of Stateless Web Services, *In Proceedings of ECOWS'06, IEEE Computer Society*, Washington, DC, pages 75–86, 2006.
3. H N Talantikite. Semantic Annotations for Web Services Discovery and Composition, Computer Standards Interfaces, 31(6):1108–1117. Elsevier B V, 2009.
4. I B Arpinar. Ontology-driven Web Services Composition Platform, *Inf. Syst. E-Business Management*, 3(2):175–199, 2005.
5. J Gekas and M Fasli. Automatic Web Service Composition Based on Graph Network Analysis Metrics, *In Proceedings of the International Conference on Ontology, Databases and Applications of Semantics (ODBASE). Agia Napa, Cyprus*, pages 1571–1587, 2005.
6. J Kwon, K Park, D Lee and S Lee. PSR: Precomputing Solutions in RDBMS for Fast Web services Composition Search, *In: Proceedings of the* 2^{nd} *International Conference on Web Services, Salt Lake City, Utah, USA*, pages 808–815, 2007.
7. Lee D, Kwon J, Lee S and Park S Hong B. Scalable and Efficient Web Services Composition based on a Relational Database, *Journal of Systems and Software*, 84(12):2139–2155, 2011.
8. C Zheng, W Ou, Y Zheng and D Han. Efficient Web Service Composition and Intelligent Search Based on Relational Database. *International Conference on Information Science and Applications (ICISA)*, pages 1–8, 2010.

Lakshmi H N is currently a Ph.D scholar, SCIS(School of Computer and Information Sciences), University of Hyderabad, Hyderabad. She is working as Associate Professor, CVR College of Engineering, Hyderabad. She received her MS in Software Systems from BITS Pilani and B.Tech degree from BMS College of Engineering, Bangalore University. Her areas of interest include Web Services and Data Structures.

Hrushikesha Mohanty is a Professor in SCIS (School Of Computer and Information Sciences), University of Hyderabad, India. Worked previously in Electronics Corporation of India Limited, Hyderabad. Took M.Sc(Mathematics) at Utkal University and completed his Ph.D(Computer Science) from IIT Kharagpur. He has over 85 research publications in refereed International Journals and Conference Proceedings.