Advanced Science and Technology Letters
Vol.82 (SoftTech 2015), pp.105-110
http://dx.doi.org/10.14257/astl.2015.82.20

Towards Service Composition Based on Theorem Proving

Suichu Zhai[1], Zhong Xin[2], and Aihua Song[3]

[1]Hangzhou Power Supply Cooperation, Hangzhou, China
[2]Jilin Technology College of Electronic Information, Jilin City, China
[3] Hangzhou Dianzi University, Hangzhou, China
yinyuyu@hdu.edu.cn

Abstract. Web Service composition has become a critical technology to react to the complex functional requirements put forward by business logic in e-commerce. How to guarantee the reliability of composite Web service is a hot topic through all the phase of lifecycle, including design, development and deployment. To this end, we are motivated to propose a Type Theory (TT) based method to automatic service composition. First of all, a service model in form of TT representation defines the atomic services declarations, semantic relation and composition request. Then, a set of TT rules that helps to prove the satisfiability of the user's request is formally given. Later, several useful tools transforming the service declaration files to the propositions and extracting the composition result from the proof is discussed. To face the challenge of reliability, we show a hybrid model, which dynamically binds service with the corresponding strict proving, to ensure the reliability of our method during compositing services. At last, the proposed method is implemented and a complete case study is conducted to demonstrate the applicability and effectiveness of our techniques.

Keywords: web service composition, composition engine, type theory

1 Introduction

Service-Oriented Computing (SOC) is a set of basic principles for designing and developing software in services industry. Each service is well-defined with business functionality which exposes its input and output information for potential invocations. Presently, there are many platforms and languages had proposed in literatures that allow to easily use heterogeneous Web Services online. The standards, such as Universal Description Discovery and Integration (UDDI), Web services Description Language (WSDL), Simple Object Access Protocol (SOAP) and Ontology Web Language for Services (OWL-S), have defined the ways to service description, selection and composition. However, service composition is used mainly to satisfy the complex requirement by composing some services together, including atomic service or composite services. The most promising method is called as Business Process Execution Language for Web Service (BPEL4WS) which focuses on representing composite service.

ISSN: 2287-1233 ASTL

A large number of service composition methods have been proposed. They can be divided into three classes: workflow, AI-planning and theorem-proving. Manual operations are needed in workflow, especially in modeling step, namely workflow methods are not automatic. In the contract, the AI-planning methods and theorem-proving methods are automatic. For the former, the AI-planning method, however, performs badly when mass services involved because of the state-explosion. For the latter, the most theorem-proving methods perform better in reliability. Nevertheless, most AI-planning and theorem-proving methods can handle the semantic information of interfaces outside of the logic framework, which reduces reliability.

In this paper, a web service composition method of theorem-proving is proposed, based on Type Theory (for short, TT). The main process of our method is transforming atomic services descriptions and semantic relation of interfaces into propositions in the form of axioms in TT. And the specification of user's requirement is formatted as conjecture expression. The way to compose web service request is to prove the conjecture by the previously mentioned propositions. In summary, this process of transforming and proving are automatic.

Our method faces the challenge of reliability in transforming the semantic relations and services declarations into propositions, then proving the conjecture requirement representation, automatically in TT. Both of the semantic relations and service connections have been formalized and proved in this process.

The three main contributions of our work are listed as follows.

(1) We propose a reliable automatic method using TT Prover to compose web service. Our method faces the challenges of reliability by proving the corresponding conjecture of the requirement strictly. Our research extends the research scope of TT.

(2) A hybrid service model is proposed in out method, which binds the concepts of web service, the logic proposition on type in TT. This model bridges proposition-proving and service composition. The model is different from other theorem-proving methods since any operation on it (including checking semantic similarity) must be formalized and proved, which guarantees the reliability of our method. What's more, this model can be used in not only service composition but also service selection and recommendation.

(3) We implement the method and build a complete system with lots of tools. The system can be used and extended in many research and application context.

The remainder is organized as follows: Section II summarizes the related works. Section III introduces the architecture of the system. Section IV introduces main technologies. Section V shows a case study and experiments. Section VII draws a conclusion and future research directions.

2 Related Work

Different researchers have different understandings of service composition. [1] takes the view of workflow. Service composition is considered as connecting services together using some special rules to satisfy a business requirement. [2] takes the view of program synthesis. Service composition is understood as integrating enterprise information systems (e.g., ERP, OA, etc.) and softwares. In [3], service composition

is thought as finding a combination of some exciting services to satisfy the request of the user from the point of problem solving. However, in [4], service composition is thought as decomposing an entire task to some sub-tasks and finding each solution for these subtasks from the point of task planning.

The methods of existing models vary widely from one to another. Depending on the composition scheme generation way, web service composition methods can be divided into three categories: Workflow; AI-planning and Theorem-Proving.

Workflow service composition is based on traditional workflow technology. The difference between workflow and service composition is analyzed in [1] [5]. The likeness of workflow and service composition is that they both have a similar life cycle, which means they both have the modeling step and the running step. Data stream and control stream are need to be designed in the modeling step. An engine analyses the definition of the data stream and control stream to build a runnable-instance in running step. However, there is still a huge gap between workflow and service composition. The atomic applications defined in the workflow are static, which means they will not be modified or removed frequently. The atomic services, however, are dynamic [6] [7]. There are many famous techniques including eFlow [8] [9] and PPM [10]. These methods are not automatic, since the help of user is needed in modeling step.

AI-planning methods takes the view of problem solving. The service composition problem is considered as an automated solving of the planning problem. Once a special initial state and a target state are clearly defined, the composition process is finding a suitable path from the initial state to the target state. PDDL [11] and Golog [12] are famous AI-planning method. Mayer in [13] presents a consistency based service composition approach, where service composition problems are modeled as the Constraint Satisfaction Problem (CSP). A composition framework named CWSF (Composition Web Service Framework), which is also based on CSP, is proposed in [14]. The composition methods based on heuristic search are mentioned in [15]. A model, called Service dependency graph (SDG) allows modeling of complex and application-specific interfaces, which is presented in [16].

Theorem-proving, however, goes another way. Even though efficiency is the most difficult problem of all theorem-proving methods, many researchers are still doing more jobs in this field because of its relibility. One of the improvements is the method based on Linear Logic (LL), proposed by Rao [17].

3 Architecture

The architecture of our method is shown as Figure 1. The basic compositions of this system are as follows:

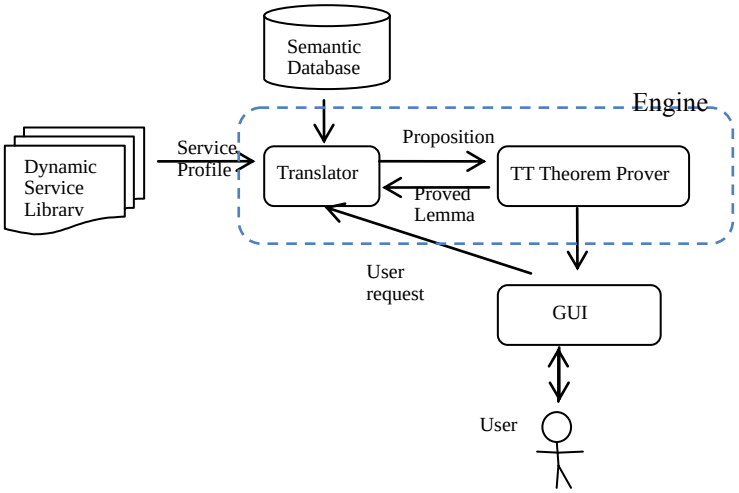

Figure 1. Architecture

Dynamic Service Library stores web services declarations. The Dynamic Service Library is used to provide necessary web services to satisfy the query of the user. The services are represented in the form of OWL-S (Ontology Web Language for Services). This library, however, is not static. It is easy to add and delete web services dynamically.

Translator is a critical component. Mainly, it has three functions. First, it transfers services profiles, which are in the form of OWL-S, provided by Dynamic Service Library into propositions in the form of expressions in TT. Second, it transfers the semantic relations of the interfaces into propositions. Third, it transfers the proved conjecture into BPEL4WS after proving.

Semantic Database is used to provide semantic ontology to analyze service profiles. The port of a web service can be defined as a concept, which is a node of the ontology tree. The subtype relation and belonging relation are easy to be found by searching the semantic ontology tree. Subtype relation and belonging relation are both used as propositions to prove targeted conjecture

TT Theorem Prover is the core component of this system. Essentially, it works as a theorem prover. Different prove tactics (i.e. Distribution rate of conjunction) and existing proposition (i.e. Atom Web Services) will be used to prove the conjecture. If the proof exists, the composition plan is contained in this proof. The proof will be transferred into a BPEL4WS file by the translator. If the proof doesn't exist, it means the composition request cannot be satisfied in current service context. Coq is used as the proving-tool in the prover.

GUI helps user typing in the query specification into the system. The composition result is also shown by the GUI.

A simple composition process is as follows. At first, user types in request specification with the help of GUI. Then the specification is transferred to a

conjecture. The Dynamic Service library provides corresponding atomic services in the form of OWL-S. The translator transforms these atomic services to propositions. Subtype and Belonging relations are also transformed into propositions with the help of Semantic Database. TT prover tries to prove the conjecture using the propositions. If success, the result will be converted into BPEL4WS by Translator. Otherwise, the failure message will be shown by the GUI.

References

1. Paolucci, Massimo, Kawamura, Takahiro, Payne, Terry R., Sycara, Katia. "Semantic matching of web services capabilities." The Semantic Web—ISWC 2002. (2002) June 333-347; Sardinia, Italy.
2. Srinivasan, Naveen, Massimo Paolucci, and Katia Sycara. "Adding OWL-S to UDDI, implementation and throughput." proceeding of Semantic Web Service and Web Process Composition 2004 (2004).
3. Zhou, Chen, Liang-Tien Chia, Bu-Sung Lee. "Service discovery and measurement based on DAML-QoS ontology." Special interest tracks and posters of the 14th international conference on World Wide Web. ACM, (2005).
4. Majithia, Shalil. "Reputation-based semantic service discovery." Enabling Technologies: Infrastructure for Collaborative Enterprises, 2004. WET ICE 2004. 13th IEEE International Workshops on. IEEE, (2004).
5. Bucchiarone, Antonio, and Stefania Gnesi. "A survey on services composition languages and models." International Workshop on Web Services–Modeling and Testing (WS-MaTe 2006). (2006).
6. Syu, Yang, et al. "Towards a genetic algorithm approach to automating workflow composition for web services with transactional and qos-awareness." Services (SERVICES), 2011 IEEE World Congress on. IEEE, (2011).
7. Blake, M. Brian, and David J. Cummings. "Workflow composition of service level agreements." Services Computing, 2007. SCC 2007. IEEE International Conference on. IEEE, (2007).
8. Casati, Fabio, et al. "Adaptive and dynamic service composition in eFlow." Advanced Information Systems Engineering. Springer Berlin/Heidelberg, (2000).
9. Casati, Fabio, et al. "eFlow: a platform for developing and managing composite e-services." Research Challenges, 2000. Proceedings. Academia/Industry Working Conference on. IEEE, (2000).
10. Schuster, Hans, et al. "Modeling and composing service-based and reference process-based multi-enterprise processes." Advanced Information Systems Engineering. Springer Berlin/Heidelberg, 2000.
11. McDermott, Drew. "Estimated-regression planning for interactions with web services." Proc. AIPS. Vol. 2. 2002.
12. McIlraith, Sheila, and Tran Cao Son. "Adapting golog for composition of semantic web services." PRINCIPLES OF KNOWLEDGE REPRESENTATION AND REASONING-INTERNATIONAL CONFERENCE-. Morgan Kaufmann Publishers; 1998, (2002).
13. Mayer, Wolfgang, Rajesh Thiagarajan, Markus Stumptner. "Service composition as generative constraint satisfaction." Web Services, 2009. ICWS 2009. IEEE International Conference on. IEEE, (2009).
14. Karakoc, E., and P. Senkul. "Composing semantic Web services under constraints." Expert Systems with Applications. 36 8 (2009).
15. Meyer, Harald, and Mathias Weske. "Automated service composition using heuristic search." Business Process Management (2006): 81-96.

16. Syu, Yang, et al. "An automated workflow composition to semantic web services." Machine Learning and Cybernetics (ICMLC), 2011 International Conference on. Vol. 2. IEEE, (2011).
17. Rao, Jinghai, and Peep Küngas. "Application of linear logic to web service composition." In The First International Conference on Web Services, Las Vegas. (2003).

