Automated Synthesis of Composite Services with Correctness Guarantee

Ting Deng, Jinpeng Huai, Xianxian Li, Zongxia Du, Huipeng Guo
School of Computer Science & Engineering, Beihang University
Beijing, CHINA
{dengting, huaijp, lixx, duzx, guohp}@act.buaa.edu.cn

ABSTRACT

In this paper, we propose a novel approach for composing existing web services to satisfy the correctness constraints to the design, including freeness of deadlock and unspecified reception, and temporal constraints in Computation Tree Logic formula. An automated synthesis algorithm based on learning algorithm is introduced, which guarantees that the composite service is the most general way of coordinating services so that the correctness is ensured. We have implemented a prototype system evaluating the effectiveness and efficiency of our synthesis approach through an experimental study.

Categories and Subject Descriptors

H.3.5 [**Information Storage and Retrieval**]: On-line Information Services—*Web-based services*

General Terms

Algorithm, Design

Keywords

composition synthesis, correctness constraints, learning algorithm.

1. INTRODUCTION

Automated service composition issue has emerged as an important and challenging problem in Web service applications, which is concerned with combining existing web services when a client request cannot be satisfied by any individual service[5].

Most of automated service composition approaches require developers to provide a detailed specification of the desired behaviors of a composite service (e.g., goal service in [1] or conversation protocol in [4]) with formal models or a specification language (e.g, BPEL4WS). To ensure the correctness of the design, developers of a composite service need to perform formal verification of the correctness constraints such as freeness of deadlock and unspecified reception, realizability[4] and temporal properties[4], etc. The process of design, verification, analysis, error detection and correction makes the composition synthesis a difficult and time-consuming task.

In this paper we propose a novel approach to synthesize the composite service from a given set of services, where the designer only needs to set the correctness constraints on the desired behaviors of the targeted service and the synthesis will be automatically performed with the correctness guaranteed. We implemented a prototype system and the preliminary experimental results on a practical travel agent example show that our synthesis approach is effective and efficient.

2. COMPOSITION MODEL

We specify a *service* in the level of business protocol or abstract business process in BPEL4WS by using deterministic finite automata (DFA), where states represent the different phases when it interacts with a user and transitions are triggered by its input/output messages. A *composite service* M which coordinates available services by routing the output messages of one service to the input messages of another is also modeled as DFA. We define the *interaction* $M||\boldsymbol{S}$ between composite service M and services in set $\boldsymbol{S}$ their synchronization on input and output actions.

Besides the conventional *correctness constraints* such as freeness of deadlock and unspecified receptions, we consider temporal properties specifying behavior of services. Specifically, we consider correctness constraints which can be represented as conjunction of two CTL formulas for safety properties and liveness properties, respectively, denoted by $\varphi_{safe} \wedge \varphi_{live}$. Due to the space limitations, readers are referred to [3] for details of CTL logic. As we know, the safety properties and liveness properties are two kinds of important properties in system verification [6]. For example, the CTL formula $\mathbf{AG}\ P$ (meaning "along every computation all states satisfy P") is a safe property, while $\mathbf{EF}\ P$ (meaning "there is a computation along which some state satisfy P") is a liveness property.

Our *composition synthesis problem* is to generate a composite service M from a given set $\boldsymbol{S}$ of services and correctness constraints $\varphi_{safe} \wedge \varphi_{live}$ in CTL logic formula such that $M||\boldsymbol{S}$ is free of deadlock and unspecified receptions, and $M||\boldsymbol{S} \models \varphi_{safe} \wedge \varphi_{live}$. We denote such composite service by $M(\boldsymbol{S}, \varphi_{safe} \wedge \varphi_{live})$.

3. AUTOMATED SYNTHESIS APPROACH

As shown in Figure 2, we propose an automated synthesis approach which is divided into three steps.

Firstly, we consider synthesizing a *most general safe* composite service M_{safe} such that $M_{safe}||\boldsymbol{S} \models \varphi_{safe}$ and $L(M')$ $\subseteq L(M_{safe})$ for every safe composite service M' (i.e., $M'||\boldsymbol{S}$

$\models \varphi_{safe})$, where $L(M)$ denotes the language accepted by M. We know that M_{safe} is a prefix-closed regular language because φ_{safe} is a safety property.

We use learning approach, specifically the L^* algorithm in [7], to synthesize M_{safe}. Figure 1 shows the overview of L^* algorithm (Due to space limitations, we do not present the algorithm in detail). L^* algorithm is a designed for learning an unknown regular language U (in our setting, the $L(M_{safe})$), by constructing a minimal DFA C (in out setting, the M_{safe}) such that $L(C) = U$. In this algorithm, a Learner asks two kinds of questions: membership queries and equivalence queries, and infers the C based on answers of teachers. Here the membership query asks whether a string σ is in U and the equivalence query for a candidate DFA C asks whether C is equivalent with U. An attractive characteristic of L^* algorithm is that the number of queries is polynomial in the size of final DFA C. Therefore in our composition synthesis problem, L^* algorithm is highly desired when the size of M_{safe} is much smaller than the size of whole state space in composition.

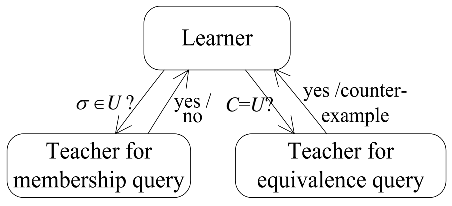

Figure 1: Overview of L* Algorithm

The second step is to deal with deadlocks in $M_{safe}||\boldsymbol{S}$. We design an algorithm to determine whether a deadlock state can be removed. The algorithm is based on an observation that if there exists a action sequence σ of M_{safe} composed of only input actions, then $M_{safe}||\boldsymbol{S}$ will reach deadlock state if services in $\boldsymbol{S}$ send messages consistent with σ.

Finally, when $M_{safe}||\boldsymbol{S}$ satisfies the liveness property φ_{live} we get the solution of our composition synthesis problem.

Following the whole synthesis process, the composition synthesis fails when there exists a deadlock state can not be removed or the liveness property verification fails.

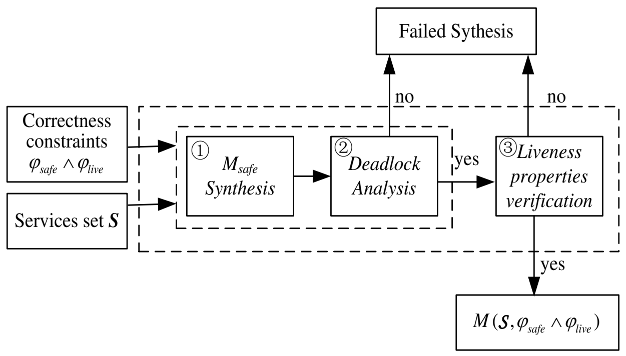
Failed Sythesis
no
no
Correctness constraints $\varphi_{safe} \wedge \varphi_{live}$
Services set S
① M_{safe} Synthesis
② Deadlock Analysis
yes
③ Liveness properties verification
yes
$M(\boldsymbol{S}, \varphi_{safe} \wedge \varphi_{live})$

Figure 2: The Synthesis Approach

4. EXPERIMENTAL RESULTS

We have implemented a prototype system to verify our synthesis approach in which a model checker NuSMV [2] is employed to answer queries in L^* algorithm. The preliminary experiments focus on the efficiency of synthesis of most safe composite service because the runtime of algorithms for removing deadlocks and checking the liveness properties is neglectable compared with one of L^* algorithm.

We use a practical Travel Agency example. A travel agency `TA` offers vacation packages to users by composing an existing flight booking service `FB` and a hotel booking service `HB`. An example of correctness constrain is that the user must book both flight and hotel or book nothing when there is no available flight or hotel, or the user doesn't like the offer. Due to space limitations, we do not present the DFA models of services and the correct constraints in CTL logic formula. The most general safe `TA` is generated by using L^* algorithm with 1284 quries where the NuSMV tool was invocated 271 times, and the execution time of L^* algorithm is about 86.39 seconds. In the realistic travel agent example, our automated synthesis approach has shown to be feasible.

5. CONCLUSION

We have proposed a novel technique for automatically synthesizing a composite service from a given services set and correctness constraints on control flow and dataflow. The technique is based on learning algorithm. We also implemented a prototype system and the preliminary experimentation shows promising results. In future work, we plan to improve our prototype system and perform more experiments to further evaluate our framework. Moreover, we will analyze the cause of failure in synthesis and design the approach in assisting the developer to perform resynthesis based on the analysis results to improve the successful rate of synthesis.

6. ACKNOWLEDGEMENT

This work is partially supported by grants from the China 863 High-tech Program (2006AA01Z19A, 2007AA010301, 2007AA01Z426), China 973 Fundamental R&D Program (2005CB321803), National Natural Science Funds for Distinguished Young Scholar (60525209) and Program for New Century Excellent Talents in University (NCET-05-0186).

7. REFERENCES

[1] D. Berardi, D. Calvanese, G. De Giacomo, R. Hull, and M. Mecella. Automatic composition of transition-based semantic web services with messaging. In *VLDB*, pages 613–624, 2005.

[2] A. Cimatti, E. M. Clarke, E. Giunchiglia, F. Giunchiglia, M. Pistore, M. Roveri, R. Sebastiani, and A. Tacchella. Nusmv 2: An opensource tool for symbolic model checking. In *CAV '02*, pages 359–364, 2002.

[3] E. M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking*. MIT Press, 2000.

[4] X. Fu, T. Bultan, and J. Su. Analysis of interacting bpel web services. In *WWW*, pages 621–630, 2004.

[5] R. Hull and J. Su. Tools for composite web services: a short overview. *SIGMOD Rec.*, 34(2):86–95, 2005.

[6] P. Manolios and R. Trefler. Safety and liveness in branching time. In *LICS '01*, pages 366–374, 2001.

[7] R. L. Rivest and R. E. Schapire. Inference of finite automata using homing sequences. *Inf. Comput.*, 103(2):299–347, 1993.