Special Focus on Information Fusion

Guest Editors: JING ZhongLiang, LEUNG Henry, LI YuanXiang
## Editorial Board

### Honorary Editor General
ZHOU GuangZhao (Zhou Guang Zhao)

### Editor General
ZHU ZuoYan Institute of Hydrobiology, CAS

### Editor-in-Chief
LI Wei Beihang University

### Executive Associate Editor-in-Chief
WANG DongMing Centre National de la Recherche Scientifique

### Associate Editors-in-Chief

<table>
<thead>
<tr>
<th>Name</th>
<th>Institution</th>
</tr>
</thead>
<tbody>
<tr>
<td>GUO Lei</td>
<td>Academy of Mathematics and Systems Science, CAS</td>
</tr>
<tr>
<td>HUANG Ru</td>
<td>Peking University</td>
</tr>
<tr>
<td>QIN YuWen</td>
<td>National Natural Science Foundation of China</td>
</tr>
<tr>
<td>SUN ZengQi</td>
<td>Tsinghua University</td>
</tr>
<tr>
<td>YOU XiaoHu</td>
<td>Southeast University</td>
</tr>
<tr>
<td>ZHAO QinPing</td>
<td>Beihang University</td>
</tr>
<tr>
<td>ZHAO Wei</td>
<td>University of Macau</td>
</tr>
</tbody>
</table>

### Members

<table>
<thead>
<tr>
<th>Name</th>
<th>Institution</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHEN JianEr</td>
<td>Texas A&amp;M University</td>
</tr>
<tr>
<td>DU Richard LiMin</td>
<td>Voiceeasy Institute of Technology</td>
</tr>
<tr>
<td>GAO Wen</td>
<td>Peking University</td>
</tr>
<tr>
<td>GE ShuZhi Sam</td>
<td>National University of Singapore</td>
</tr>
<tr>
<td>GUO GuangCan</td>
<td>University of Science and Technology of China</td>
</tr>
<tr>
<td>HAN WenBao</td>
<td>PLA Information Engineering University</td>
</tr>
<tr>
<td>HE JiFeng</td>
<td>East China Normal University</td>
</tr>
<tr>
<td>HU WeiWu</td>
<td>Institute of Computing Technology, CAS</td>
</tr>
<tr>
<td>HU ZhanYi</td>
<td>Institute of Automation, CAS</td>
</tr>
<tr>
<td>IDA Tetsuo</td>
<td>University of Tsukuba</td>
</tr>
<tr>
<td>JI YueFeng</td>
<td>Beijing University of Posts and Telecommunications</td>
</tr>
<tr>
<td>JIN Hai</td>
<td>Huazhong University of Science and Technology</td>
</tr>
<tr>
<td>JIN YaQiu</td>
<td>Fudan University</td>
</tr>
<tr>
<td>JING ZhongLang</td>
<td>Shanghai Jiao Tong University</td>
</tr>
<tr>
<td>LI Joshua LeWei</td>
<td>University of Electronic Science and Technology of China</td>
</tr>
<tr>
<td>LIU DeRong</td>
<td>Institute of Automation, CAS</td>
</tr>
<tr>
<td>LIN HuiMin</td>
<td>Institute of Software, CAS</td>
</tr>
<tr>
<td>LIN ZongLi</td>
<td>University of Virginia</td>
</tr>
<tr>
<td>LONG KePing</td>
<td>University of Science and Technology Beijing</td>
</tr>
<tr>
<td>LU Jian</td>
<td>Nanjing University</td>
</tr>
<tr>
<td>MEI Hong</td>
<td>Peking University</td>
</tr>
<tr>
<td>MENG LuoMing</td>
<td>Beijing University of Posts and Telecommunications</td>
</tr>
<tr>
<td>PENG LianMao</td>
<td>Peking University</td>
</tr>
<tr>
<td>PENG QuSheng</td>
<td>Zhejiang University</td>
</tr>
<tr>
<td>SHEN ChangXiang</td>
<td>Computing Technology Institute of China Navy</td>
</tr>
<tr>
<td>SUN JiaGuang</td>
<td>Tsinghua University</td>
</tr>
<tr>
<td>TANG ZhiMin</td>
<td>Institute of Computing Technology, CAS</td>
</tr>
<tr>
<td>TIAN Jie</td>
<td>Institute of Automation, CAS</td>
</tr>
<tr>
<td>TSAI WeiTek</td>
<td>Arizona State University</td>
</tr>
<tr>
<td>WANG Ji</td>
<td>National University of Defense Technology</td>
</tr>
<tr>
<td>WANG JiangZhou</td>
<td>University of Kent</td>
</tr>
<tr>
<td>WANG Long</td>
<td>Peking University</td>
</tr>
<tr>
<td>WU YiRong</td>
<td>Institute of Electronics, CAS</td>
</tr>
<tr>
<td>XIE WeiXin</td>
<td>Shenzhen University</td>
</tr>
<tr>
<td>XU Jun</td>
<td>Tsinghua University</td>
</tr>
<tr>
<td>XU Ke</td>
<td>Beihang University</td>
</tr>
<tr>
<td>YIN QinYe</td>
<td>Xi’an Jiaotong University</td>
</tr>
<tr>
<td>YING MingSheng</td>
<td>Tsinghua University</td>
</tr>
<tr>
<td>ZHA HongBin</td>
<td>Peking University</td>
</tr>
<tr>
<td>ZHANG HuanGuo</td>
<td>Wuhan University</td>
</tr>
<tr>
<td>ZHOU Dian</td>
<td>The University of Texas at Dallas</td>
</tr>
<tr>
<td>ZHOU ZhiHua</td>
<td>Nanjing University</td>
</tr>
<tr>
<td>ZHUANG YueTing</td>
<td>Zhejiang University</td>
</tr>
</tbody>
</table>

### Editorial Staff

<table>
<thead>
<tr>
<th>Name</th>
<th>Institution</th>
</tr>
</thead>
<tbody>
<tr>
<td>SONG Fei</td>
<td>FENG Jing</td>
</tr>
<tr>
<td>ZHAO DongXia</td>
<td></td>
</tr>
</tbody>
</table>
Contents

Special Focus on Information Fusion

On the sensor order in sequential integrated probability data association filter................................................................. 491
WANG Yang, JING ZhongLiang, HU ShiQiang, ZHANG HongJian & WU JingJing

Joint spatial registration and multi-target tracking using an extended PM-CPHD filter....................................................... 501
LIAN Feng, HAN ChongZhao, LIU WeiFeng, LIU Jing & YUAN XiangHui

Globally optimal distributed Kalman filtering fusion ........................................................................................................... 512
SHEN XiaoJing, LUO YingTing, ZHU YunMin & SONG EnBin

Data fusion for target tracking in wireless sensor networks using quantized innovations and Kalman filtering.................. 530
XU Jian, LI JianXun & XU Sheng

Integrated optimization methods in multisensor decision and estimation fusion ................................................................. 545
LUO YingTing, SHEN XiaoJing & ZHU YunMin

A new DSmT combination rule in open frame of discernment and its application .............................................................. 551
WEN ChengLin, XU XiaoBin, JIANG HaiNa & ZHOU Zhe

Some notes on betting commitment distance in evidence theory .......................................................................................... 558
HAN DeQiang, DENG Yong, HAN ChongZhao & YANG Yi

A dual-kernel-based tracking approach for visual target ....................................................................................................... 566
ZHANG CanLong, JING ZhongLiang, JIN Bo & LI Zhixin

A multi-cue mean-shift target tracking approach based on fuzzified region dynamic image fusion ...................................... 577
XIAO Gang, YUN Xiao & WU JianMin

Fusion tracking in color and infrared images using joint sparse representation ................................................................. 590
LIU HuaiPing & SUN FuChun

Video motion stitching using trajectory and position similarities ........................................................................................ 600
CHEN XiaoWu, LI Qing, LI Xin & ZHAO QinPing

Pan-sharpening: a fast variational fusion approach ............................................................................................................... 615
ZHOU ZeMing, LI YuanXiang, SHI HanQing, MA Ning & SHEN Ji

An algorithm for high precision attitude determination when using low precision sensors .................................................. 626
CAO Lu, CHEN XiaoQian & SHENG Tao

RESEARCH PAPER

Complexity of synthesis of composite service with correctness guarantee ................................................................. 638
DENG Ting, HUAI JinPeng & WO TianYu

RST transforms resistant image watermarking based on centroid and sector-shaped partition ........................................... 650
ZHAO Yao, NI RongRong & ZHU ZhenFeng

CUDA-Zero: a framework for porting shared memory GPU applications to multi-GPUs .................................................. 663
CHEN DeHao, CHEN WenGuang & ZHENG WeiMin

A fast successive over-relaxation algorithm for force-directed network graph drawing ...................................................... 677
WANG YongXian & WANG ZhengHua

Achieving fair service with a hybrid scheduling scheme for CICQ switches ........................................................................ 689
HU HongChao, GUO YunFei, YI Peng & LAN JuLong

Estimation of convergence rate for multi-regression learning algorithm .............................................................................. 701
XU ZongBen, ZHANG YongQuan & CAO FeiLong

The optimization of replica distribution in the unstructured overlays .................................................................................. 714
FENG GuoFu, LI WenZhong, LU SangLu, CHEN DaoXu & BUYYA Rajkumar

A new intelligent prediction system model—the compound pyramid model ........................................................................ 723
YANG BingRu, QU Wu, WANG LiJun & ZHOU Ying
Complexity of synthesis of composite service with correctness guarantee

DENG Ting$^{1,2,*}$, HUAI JinPeng$^{1,3}$ & WO TianYu$^{1,3}$

$^1$State Key Laboratory of Software Development Environment, Beihang University Beijing 100191, China; $^2$School of Mathematics and System Science, Beihang University, Beijing 100191, China; $^3$School of Computer Science and Engineering, Beihang University, Beijing 100191, China

Received December 13, 2010; accepted May 2, 2011

Abstract How to compose existing web services automatically and to guarantee the correctness of the design (e.g. temporal constraints specified by temporal logic LTL, CTL or CTL*) is an important and challenging problem in web services. Most existing approaches use the process in conventional software development of design, verification, analysis and correction to guarantee the correctness of composite services, which makes the composition process both complex and time-consuming. In this paper, we focus on the synthesis problem of composite service; that is, for a given set of services and correctness constraint specified by CTL or CTL* formula, a composite service is automatically constructed which guarantees that the correctness is ensured. We prove that the synthesis problem for CTL and CTL* are complete for EXPTIME and 2EXPTIME, respectively. Moreover, for the case of synthesis failure, we discuss the problem of how to disable outputs of environment (i.e. users or services) reasonably to make synthesis successful, which are also proved complete for EXPTIME and 2EXPTIME for CTL and CTL*, respectively.

Keywords business protocol, composite service, synthesis, environment, branching temporal logic

1 Introduction

Service-oriented computing has become a promising paradigm for realizing distributed applications, as more and more web services are being developed and published on the Internet based on SOAP$^1$, WSDL$^2$ and BPEL$^3$. These services can serve as the reusable components for building complex applications. Recently, the web service composition issue has emerged as an important and challenging problem in Web service applications, which is concerned with how to combine existing web services when a client request cannot be satisfied by any individual service [1].

There have been many efforts towards automated service composition, and most of them are based on formal methods including automata theory, logical reasoning, planning in AI and theorem proving [1–6]. Most of these approaches require developers to provide a detailed specification of the desired behaviors
of a composite service (e.g., goal service in [2,3] or conversation protocol in [6]) 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 temporal properties [7,8].

The process of design, verification, analysis and correction makes the composition synthesis a difficult and time-consuming task. So in this paper, we focus on the problem of synthesizing the composite service from the library of services such that correctness constraints specified by temporal logic formulas are guaranteed during the synthesis process.

Synthesis is the automated construction of a system from its specification. In the literature, a variety of synthesis problem have been studied for closed system [9,10] and open systems [11–14]. A closed system is a system whose behavior is completely determined by its own state. An open system is one that interacts with its environment and whose behavior crucially depends on this interaction. Since services interact with users and other services through message exchanges, they can be treated as open systems taking users or services as their environments. In [11–14], systems are constructed from scratch instead of composing from reusable components. However, almost every non-trivial system, especially the Web-based system, is constructed using libraries of reusable components. In [15], the synthesis from component library for LTL was studied in which only linear temporal properties are guaranteed.

In light of these, we investigate the synthesis of composite service from library of services for CTL and CTL*, and establish their respective complexity bounds. CTL and CTL* are two kinds of branching temporal logic which can support both branching and linear temporal properties. Furthermore, for the case of synthesis failure; that is, when there exists no composite service over given library of services which satisfies given CTL/CTL* formula, we observe that we can still construct the desired composite service by restricting environment’s output behaviors reasonably, which is referred to as the environment-controllable synthesis. Correspondingly, the former synthesis of composite service is referred to as the environment-uncontrollable synthesis. We give formal treatments of these two synthesis problem and establish the complexity bounds for them for CTL and CTL*, respectively.

The paper is organized as follow. Section 2 discusses related work. Section 3 gives some preliminary definitions. Section 4 and Section 5 give formal investigation of the environment-uncontrollable synthesis problem and environment-controllable synthesis problem, respectively. Section 6 concludes the work.

2 Related work

Several approaches have been proposed to automated synthesis of composite services. The Roman model specified the goal service and e-services as finite state machine and reduced the composition synthesis into satisfiability problem of PDL formula [2]. The Colombo model extended [2] with message passing and world state of local database [3]. Fan et al. [4] modeled the goal service and services as alternating finite automata and discussed the time complexity of composition synthesis by exploring connections between composition synthesis and query rewriting using view. Mitra et al. [5] modeled services and goal service as I/O automata, and reduced the existence problem of choreography to the simulation of I/O automata. Fu et al. [6] modeled global behavior of e-service composition in asynchronous messages as conversation based on Mealy machine and discussed the realizability and synchronization of conversations.

Above researches all require developer to provide a priori detailed behavior specification for composite service. In contrast to researches above, we only require developer to provide the correctness constraint on composition and automatically synthesize the behavior of composite service.

Huai et al. [16] and Pistore et al. [17] presented composition framework AutoSyn and Astro, and studied mediator-based synthesis of composite service from CTL formula and EAGLE formula, respectively. In contrast, the composite scenario in this paper is not mediator-based, which means that composite service can be invoked as a component by another composite service. Moreover, in order to handle composition failure, we study the restriction mechanism to environment behaviors which is not considered in [16,17].

In addition, there are other papers on automated service composition from semantic or syntactic description of services which consider a service as function with input and output, and use the classical
planning approach in AI [18,19] or type derivation algorithm [20]. In this paper, we don’t consider the semantic description of services, but services are characterized as more general transducer model which has the function with input and output as a special case.

Synthesis of open systems (or reactive systems) has been explored in many research work. The synthesis problem of single reactive systems with synchronous and asynchronous behaviors for LTL is studied in [11,12], respectively. The complexity bounds for synthesis problem for CTL and CTL* with either complete or incomplete information are provided in [13]. In addition, the synthesis of distributed systems for CTL*, LTL and CTL are investigated in [14]. All of these constructed system from scratch rather than composing from reusable components. In addition, they also didn’t consider restricting environment behaviors to handle composition failure like we do.

Closer to this work are [15] which studied the synthesis from component library for LTL. Since LTL is linear, it should be hold by every execution path generated by the composite services, so it doesn’t work for branching time properties hold by some execution paths of composite services. Thus we study synthesis from component library for branching time logic including CTL and CTL*. Also the restriction mechanism on environment was not considered in [15].

3 Preliminaries

Given a set $D$ of directions, a $D$-tree is a set $T \subseteq D^*$ such that if $x \cdot c \in T$ where $x \in D^*$ and $c \in D$, then also $x \in T$. The elements of $T$ are called nodes, and the empty word $\varepsilon$ is the root of $T$. For every $x \in T$, all the nodes $x \cdot c$, where $c \in D$, are the successors of $x$. The number of successors of $x$ is called the degree of $x$ and is denoted by $d(x)$. A tree $T$ with branching degree bounded by $k$ is a tree such that the degree of every node $x$ in $T$ is $k$. A node $X$ is a leaf if it has no successors. A path $\pi$ of a tree $T$ is a set $\pi \in T$ such that $\varepsilon \in \pi$ and for every $x \in \pi$, either $x$ is a leaf or there exists a unique $c \in D$ such that $x \cdot c$. Given an alphabet $\Sigma$, a $\Sigma$-labeled tree is a pair $\langle T, V \rangle$ where $T$ is a tree and $V : T \rightarrow \Sigma$ maps each node of $T$ to a letter in $\Sigma$.

A Kripke structure is tuple $K = (AP,W,R,w_{in},L)$, where AP is a set of atomic propositions, $W$ is a set of states, $R \subseteq W \times W$ is a transition relation, $w_{in}$ is an initial state, and $L : W \rightarrow 2^{AP}$ maps each state to the set of atomic propositions true in that state. For each state $w$ in $W$, we denote the set of all successors of $w$ by $\text{succ}_K(w)$, i.e. $\text{succ}_K(w) = \{w' \in W | (w,w') \in R\}$. The number of successors of $w$ is called the degree of $w$ and is denoted by $d(w)$. A Kripke structure $K = (AP,W,R,w_{in},L)$ can be viewed as a tree $(T_K, V_K)$ which is defined as follows.

1) For $x \in T_K$, $V_K(x) = w_{in}$.
2) For $y \in T_K$ with $\text{succ}_K(V_K(y)) = \{w_0, \ldots, w_m\}$, we have $y \cdot i \in T_K$ and $V_K(y \cdot i) = w_i \ (0 \leq i \leq m)$.

A nondeterministic transducer is an nondeterministic finite automaton with outputs. Formally, a nondeterministic transducer is a tuple $M = (\Sigma^I, \Sigma^O, Q, q_{in}, \delta, \gamma, F)$ where $\Sigma^I$ is a finite input alphabet, $\Sigma^O$ is a finite output alphabet, $Q$ is a finite set of states, $q_{in} \in Q$ is an initial state, $\delta : Q \times \Sigma^I \rightarrow 2^Q$ is a nondeterministic transition function, $F$ is a set of final states, and $\gamma : Q \rightarrow 2^\Sigma$ is an output function labelling states with output letters. For each state $q$ in $Q$, we denote the set of all successors of $q$ by $\text{succ}_M(q)$, i.e. $\text{succ}_M(q) = \{q' \in W | \exists \sigma \in \Sigma^I, \text{ s.t. } q' \in \delta(q)\}$. A nondeterministic transducer $M$ corresponds to a $\Sigma^I \cup \Sigma^O$-labeled $Q \times \Sigma^I$-tree $\langle T_M, V_M \rangle$, referred to as the computation tree of $M$, where

1) The root of $T_M$ is $\langle q_{in}, \varepsilon \rangle$ such that $V_M(\langle q_{in}, \varepsilon \rangle) = \gamma(q_{in})$.
2) For any node $v \in T_M$ and $\langle q, \sigma \rangle \in Q \times \Sigma^I$, we have $V_M(v \cdot (q, \sigma)) = \sigma \cup \gamma(q)$.

A transducer $M$ satisfies a CTL/CTL* formula $\varphi$ if and only if its computation tree $\langle T_M, V_M \rangle$ satisfies $\varphi$, denoted by $M \models \varphi$. For node $\langle q_{in}, \varepsilon \rangle$ we denote $\varphi$ by $\langle q_{in}, \varepsilon \rangle$. We denote $\langle q_{in}, \varepsilon \rangle \cdot (q_1, \sigma_1) \cdot \cdots \cdot (q_k, \sigma_k)$ by $\langle q_{in}, \varepsilon \rangle$. We denote $\langle q_{in}, \varepsilon \rangle \cdot (q_1, \sigma_1) \cdot \cdots \cdot (q_k, \sigma_k)$ by $\langle q_{in}, \varepsilon \rangle$.

In addition, a Nondeterministic transducer $M$ also corresponds to a Kripke structure $K_M = (\Sigma^I \cup \Sigma^O, W, w_{in}, R, L)$, where

1) The set of states $W = \{q \cdot a | \exists q' \cdot q \cdot a \}$.
2) The initial state $w_{in} = \langle q_{in}, \varepsilon \rangle$. 
3) The transition relation $R$ is defined as follows: For each pair of states $\langle q', a' \rangle, \langle q, a \rangle \in W$, we have $R((q', a'), (q, a))$ if and only if $\delta(q', a) = q$.

4) The labeling function $L$ is defined as follows: For each state $w = \langle q, a \rangle \in W$, if $w = w_{in}$, we have $L(w) = \gamma(q)$; otherwise $L(w) = \gamma(q) \cup \{a\}$.

Obviously, the computation tree of a transducer $M$ is the same as the computation tree of its Kripke structure $K_M$.

A nondeterministic tree automaton is a tuple $A = (\Sigma, D, Q, q_{in}, F)$, where $\Sigma$ is a finite input alphabet, $D$ is a set of directions, $Q$ is a finite set of states, $q_{in} \in Q$ is an initial state, $\delta : Q \times \Sigma \times D \rightarrow 2^{D \times Q}$ is a transition function, and $F$ specifies the acceptance condition that defines a subset of $Q^w$. We define several types of acceptance conditions below. A run of a nondeterministic tree automaton $A$ over a $D$–tree $(T, V)$ is a $(T \times Q)$–tree $(T_r, r)$ which satisfies the following:

1) $\varepsilon \in T_r$ and $\gamma(\varepsilon) = \langle \varepsilon, q_{in} \rangle$.

2) Let $y \in T_r$ with $r(y) = \langle x, q \rangle$ and $((c_0, q_0), \ldots, (c_{d(x)-1}, q_{d(x)-1})) \in \delta(q, V(x), d(x))$. Then for all $0 \leq i \leq d(x) - 1$, we have $y \cdot c_i \in T_r$ and $r(y \cdot c_i) = (x \cdot c_i, q_i)$.

A run $(T_r, r)$ is accepting if all its infinite paths satisfy the acceptance condition. Given a run $(T_r, r)$ and an infinite path $p \subseteq T_r$, let $\inf(p) \subseteq Q$ be a subset of $Q$ such that $q \in \inf(p)$ if and only if there are infinitely many $y \in p$ for which $r(y) \in T \times \{q\}$. We consider Büchi acceptance in which a path $p$ is accepting if and only if $F \subseteq Q$ and $\inf(p) \cap F \neq \emptyset$, and Rabin acceptance in which a path $p$ is accepting if and only if $F = \{\langle G_1, B_1 \rangle, \ldots, \langle G_m, B_m \rangle\}$ and there exists a pair $\langle \gamma_i, B_i \rangle$ in $F$ such that $\inf(p) \cap G_i \neq \emptyset$ and $\inf(p) \cap B_i = \emptyset$.

For a given set $X$, let $B^+(X)$ be the set of positive Boolean formulas over $X$ (i.e., Boolean formulas built from elements in $X$ using $\land$ and $\lor$), where we also allow the formulas true and false and, as usual, $\land$ has precedence over $\lor$. For a subset $Y \subseteq X$ and a formula $\theta$ in $B^+(X)$, we say that $Y$ satisfies $\theta$ if and only if assigning true to elements in $Y$ and assigning false to elements in $X \setminus Y$ makes $\theta$ true. An alternating tree automaton is a tuple $A = (\Sigma, D, Q, q_{in}, \delta, F)$ where $\Sigma$ is a finite input alphabet, $D$ is a set of directions, $Q$ is a finite set of states, $q_{in} \in Q$ is an initial state, $\delta : Q \times \Sigma \times D \rightarrow B^+(D \times Q)$ is a transition function, and $F$ specifies the acceptance condition that defines a subset of $Q^w$. A run of an alternating automaton $A$ over a $D$–tree $(T, V)$ is a $(T \times Q)$–tree $(T_r, r)$ and $(T_r, r)$ satisfies the following:

1) $\varepsilon \in T_r$ and $\gamma(\varepsilon) = \langle \varepsilon, q_{in} \rangle$.

2) Let $y \in T_r$ with $r(y) = \langle x, q \rangle$ and $\delta(q, V(x), d(x)) = \theta$. Then there is a (possibly empty) set $S = \{\langle c_0, q_0 \rangle, \langle c_1, q_1 \rangle, \ldots, \langle c_{d(x)-1}, q_{d(x)-1} \rangle\} \subseteq D \times Q$ such that $S$ satisfies $\theta$, and for all $0 \leq i \leq d(x) - 1$, we have $y \cdot c_i \in T_r$ and $r(y \cdot c_i) = (x \cdot c_i, q_i)$.

An alternating word automaton $A$ is a tuple $A = (\Sigma, Q, q_{in}, \delta, F)$ where $\Sigma$ is a finite input alphabet, $D$ is a set of directions, $Q$ is a finite set of states, $q_{in} \in Q$ is an initial state, $\delta : Q \times \Sigma \rightarrow B^+(Q)$ is a transition function, and $F$ specifies the acceptance condition that defines a subset of $Q^w$. A run of an alternating word automaton $A$ over an infinite word $a_0a_1 \cdots$ is a $Q$–tree $(T_r, r)$ and $(T_r, r)$ satisfies the following:

1) $\varepsilon \in T_r$ and $\gamma(\varepsilon) = q_{in}$.

2) Let $y \in T_r$ with $r(y) = \langle x, q \rangle$ and $\delta(q, a_i) = \theta$. Then there are $k$ child nodes $y_1, \ldots, y_k$ ($k \leq |Q|$) such that $\{\gamma(y_1), \ldots, \gamma(y_k)\}$ satisfies $\theta$.

4 Environment-uncontrollable synthesis of composite services

In this section, we consider the following composition scenario which is common in Web service applications [4,15]. For a composite service composed by component services, one component service starts running when it enters its initial state, then the messaging interaction between environment (i.e. users or other services) and composite service proceeds under the control of the component service until it enters its final state. At this moment, another component service enters its initial state and starts running.

Due to the nature of interaction of services, we abstract away the precise details of the services and model a service as a nondeterministic transducer in the level of business protocol [21]. Formally, A
service is a non-deterministic transducer $C = (\Sigma^I, \Sigma^O, Q, q_{in}, \delta, \gamma, F)$ where $\Sigma^I = 2^I$, $\Sigma^O = 2^O$, and $I$ and $O$ are input alphabet and output alphabet of $C$, respectively. Intuitively, $I$ and $O$ refer to the sets of all parameters in input messages and output messages of $C$, respectively. So $\Sigma^I$ is the set of input messages and $\Sigma^O$ is the set of output messages of $C$, which means that we don’t consider the orders of parameters in messages. A library $\mathcal{C}$ is simply a collection of services. Let $\mathcal{C}$ be a collection of $n$ services $C_1, C_2, \ldots, C_n$ which w.l.o.g share the same input and output alphabets $\Sigma^I$ and $\Sigma^O$.

For a composite service $CS$ composed by (part of) services in $\mathcal{C}$, when the component service $C_i$ which is currently running enters its final state, another component service $C_j$ enters its initial state and start running. Since $C_i$ may have multiple final states, it’s a key issue to determine which component should be chosen when $C_i$ enters different final states. This problem has been explored in [15] which presented the notion of interface function. Given $k$ services $C_1, C_2, \ldots, C_k$ in library $\mathcal{C}$, where $C_i = (\Sigma^I, \Sigma^O, Q, q_{in}, \delta_i, \gamma_i, F_i)$ ($i \in [1, k]$), the interface function of $C_i$ ($i \in [1, k]$) is defined as a function $f_i : F_i \rightarrow \{1, \ldots, k\}$ where $F_i$ is the set of final states of $C_i$. Intuitively, when service $C_i$ is currently running and enters a final state $q_f \in F_i$, then the component $C_{f_i(q_f)}$ enters the initial state $q_{in}^{f_i(q_f)}$ and begins to run.

The composite service with start service $C_j$ ($j \in [1, k]$) over components $C_1, C_2, \ldots, C_k$, where each $C_i$ has interface function $f_i$, is also a non-deterministic transducer $CS = (\Sigma^I, \Sigma^O, Q, q_{in}, \delta, \gamma, F)$, where

1) The set of states is $Q = \bigcup_{i=1}^{\infty}(Q_i \times \{i\})$.
2) The initial state $q_{in} = (q_{in}^0, j)$ and we refer to $C_j$ as the start service of $CS$.
3) The transition function $\delta$ is defined as follows. For state $\langle q, i \rangle \in Q$, $\delta(\langle q, i \rangle, \sigma) = \{(q', i) | q' \in \delta_i(q, \sigma)\}$ if $q \in Q \setminus F_i$ (i.e. $q$ is not a final state of $C_i$); otherwise, $\delta(\langle q, i \rangle, \sigma) = \{(q', f_i(q)) | q' \in \delta_{f_i(q)}(q_{in}^{f_i(q)}, \sigma)\}$.
4) The output function $\gamma$ is defined as follows. For every state $\langle q, i \rangle \in Q$, $\gamma(\langle q, i \rangle) = \gamma_i(q)$ if $q \in Q \setminus F_i$; otherwise, $\gamma_i(q, \sigma) = \gamma(i(q)) = \gamma_i(q) \cup \gamma_{f_i(q)}(q_{in}^{f_i(q)}).
5) The set of final states is $F = \bigcup_{i=1}^{\infty}(F_i \times \{i\})$.

**Problem statement.** In the rest of the section we shall study the following environment-uncontrollable synthesis problems, referred to as $CS(\mathcal{C}, \varphi)$. Given a library $\mathcal{C}$ of services and a temporal logic formula $\varphi$, $CS(\mathcal{C}, \varphi)$ is to determine whether or not there exist a composite service $CS$ over (part of) services in $\mathcal{C}$ such that $CS(\varphi)$.

The complexity bounds of $CS(\mathcal{C}, \varphi)$ when $\varphi$ is a LTL formula are given in [15]. Here we establish the complexity bounds of $CS(\mathcal{C}, \varphi)$ when $\varphi$ is a CTL/CTL* formula.

**Lemma 1.** $CS(\mathcal{C}, \varphi)$ can be solved in single-exponential time when $\varphi$ is a CTL formula.

**Proof.** We develop an EXPTIME algorithm for $CS(\mathcal{C}, \varphi)$. Given $\mathcal{C} = \{C_1, \ldots, C_n\}$ and $\varphi$, the algorithm first constructs $n$ alternating word automata $A_{\varphi}^i$ ($i \in [1, n]$) such that every $A_{\varphi}^i$ ($i \in [1, n]$) accepts exactly all the trees satisfying $\varphi$ in which every tree is a computation tree of a composite service with start service $C_i$ ($i \in [1, n]$). We then show that there exists a composite service over services in $\mathcal{C}$ if and only if there exists at least one $A_{\varphi}^i$ such that (the languages of) $A_{\varphi}^i$ is nonempty.

Given a CTL formula $\varphi$, we can construct in linear running time a B"uchi alternating tree automaton $A_{\varphi} = (2^{I \cup O}, cI(\varphi), s_{in}, \delta_{\varphi}, F_{\varphi})$ where $cI(\varphi)$ is the closure of $\varphi$ (see [22] for detailed definition about closure) such that $A_{\varphi}$ accept exactly the set of trees satisfying $\varphi$. As shown in [22], the product of $A_{\varphi}$ and a Kripke structure $K$ simulates a run of $A_{\varphi}$ over the computation tree $\langle T_K, V_C \rangle$ of $K$. Intuitively, in order to construct $A_{\varphi}^i$ ($i \in [1, n]$), the algorithm first transforms every service $C_i$ to its equivalent Kripke structure $K_i$, and then it takes the following step inductively.

1) Construct the product of $A_{\varphi}$ and $K_i$. We denote the product by $A_{\varphi}^i$.
2) Repeat the following process: for every state of $A_{\varphi}^i$ which is not an initial state, if this state is associated with the Kripke structure $K_j$, compute the product of $A_{\varphi}^i$ and $K_j$ from this state. The process terminates and returns $A_{\varphi}^i$ when there are no new states of $A_{\varphi}^i$.

Let $K_i = (\Sigma^I \cup \Sigma^O, W_i, w_{in}^i, R_i, B_i, i \in [1, n]$. The set of Kripke structures of services in $\mathcal{C}$ is denoted by $\mathcal{K}$. Formally, following the steps above, we get an alternating word automaton $A_{\varphi}^i$ which is defined as $A_{\varphi}^i = (\Sigma^I_{\varphi}, W \times cI(\varphi), \langle w_{in}^i, s_{in}, \delta_{\varphi}, F_{\varphi}^i \rangle$, where

1) $\Sigma^I_{\varphi} = \{b, b_1, \ldots, b_n\}$ is the finite input alphabet.
2) $W \times cl(\varphi)$ is the set of states, where $W = W_1 \cup \cdots \cup W_n$.  
3) $\langle w, s_{in} \rangle$ is the initial state, where $w_{in}$ and $s_{in}$ are initial states of $K_1$ and $A_2$, respectively.  
4) The transition $\delta^T$ is defined as follows. For state $\langle w, s \rangle \in W \times cl(\varphi)$, where $w = \langle q, \sigma \rangle$ is a state of $K_j$ and $q$ is a state of $C_j$ (that is, $\langle w, s \rangle$ is associated with a state of $K_j$).  
   a) If $q$ is not a final state of $C_j$, Let $\text{succ}_{K_j}(w) = \{w_0, \ldots, w_{d(w) - 1}\}$ and $\delta^C(s, L(w), d(w)) = \theta$. We have $\delta^T_{\varphi}(w, s, b) = \theta'$, where $\theta'$ is obtained from $\theta$ by replacing each atom $(c, s')$ by the atom $(w_c, s')$.  
   b) otherwise, for every $j \in \{1, n\}$, let $\text{succ}_{K_j}(w_{in}) = \{w_0, \ldots, w_{d(w_{in}) - 1}\}$ (i.e. the set of all successors of initial state $w_{in}$ of $K_1$) and $\delta^C(s, L_j(w) \cup L_j(w_{in}), d(w_{in})) = \theta_1 (l \in \{1, n\})$. We have $\delta^T_{\varphi}(s, L_j(w) \cup L_j(w_{in}), d(w_{in})) = \theta'_1$, where $\theta'_1$ is obtained from $\theta_1$ by replacing each atom $(c, s')$ by the atom $(w_c, s')$.  
5) $F^\varphi = W \times F_{\varphi}$ is the acceptance condition of $A^\varphi$.

We show that there exists a composite service CS over services in library $C$ such that $CS|_{=} \varphi$ if and only if there exists at least one automaton in $\{A^\varphi_1, \ldots, A^\varphi_2\}$ which is nonempty.

Assume first that there exists a composite service $CS$ over services $C_i (i \in \{1, k\})$ in $C$ with interface functions $f_i (i \in \{1, k\})$ such that $CS|_{=} \varphi$. Let the computation tree of $CS$ be $(T, V)$. Then there is $(T, V) \models \varphi$. Recall that $(T, V)$ is a $\Sigma^T$-labeled tree, where $Q = \bigcup_{i=1}^{k} (Q_i \times \{i\})$. Since $A_2$ accepts exactly all the trees satisfying $\varphi$, $A_2$ accepts $(T, V)$. Thus there exists an accepting run $(T_r, r)$ of $A_\varphi$ over $(T, V)$. Recall that $T_r$ is labeled with $T \times cl(\varphi)$. Assume w.l.o.g. that the composite service $CS$ has start service $C_1$. We construct a tree $(T_r, r')$ which is labeled with $(\Sigma^T_1)^* \times W \times cl(\varphi)$ as follows. For every node $y \in T_r$ with $r(y) = (x, s)$, where $x \in T$ and $s \in cl(\varphi)$. Note that every service $C_i (i \in \{1, n\})$ has the same computation tree with its Kripke structure $K_i$. Let $x = x' \cdot (q, \sigma)$. If $q$ is not a final state of $C_i$, there exists a state $w$ of $K_i$ such that $V(x) = L_i(w)$; otherwise, there exists a state $w$ such that $V(x) = L_i(w) \cup L_{f_i}(q_{\text{in}}(q))$. From the definition of $A^\varphi_1$, it follows that there exists a sequence $z \in (\Sigma^T_1)^*$ with length $|x|$ such that $r'(y) = (z, w, s)$.

We show that $(T_r, r')$ is an accepting run of $A^\varphi_1$. In fact, since $F^\varphi_1 = W \times F_{\varphi}$, we only need to show that $(T_r, r')$ is a run of $A^\varphi_1$; acceptance follows from the fact that $(T_r, r')$ is accepting. Consider a node $y \in T_r$ with $r(y) = (x, s)$. Assume w.l.o.g. that $w$ is a state of $K_i$. When $w$ is not a final state of $K_i$, then we have $V(x) = L_i(x)$. Since $(T_r, r)$ is a run of $A_\varphi$, there exists a set of $\{\langle c_0, s_0 \rangle, \langle c_1, s_1 \rangle, \ldots, \langle c_{k-1}, s_{k-1} \rangle\}$ satisfying $\varphi$ such that $y$ has successors $y \cdot c_i$ and $r(y \cdot c_i) = (x \cdot c_i, s_i)$ ($i \in \{0, k - 1\}$). In $(T_r, r')$, by its definition, there exists $z \in (\Sigma^T_1)^*$ with length $|y|$ and its $k$ successor $z \cdot z_i$ where $z_i \in \Sigma^T_1 (i \in \{0, k - 1\})$ such that $r'(y) = (z, w, s)$ and $r'(y \cdot c_i) = (z \cdot z_i, w_{c_i}, s_i)$ ($i \in \{0, k - 1\}$). Let $\delta^\varphi_1(s, a) = \theta'$. By the definition of $\delta^\varphi_1$, the set $\{\langle w_{c_0}, q_0 \rangle, \langle w_{c_1}, q_1 \rangle, \ldots, \langle w_{c_{k-1}}, q_{k-1} \rangle\}$ satisfies $\theta'$. When $w$ is a final state of $K_i$, we can get the same result like above. Thus $(T_r, r')$ is a run of $A^\varphi_1$.

Assume w.l.o.g. that $A^\varphi_2$ is nonempty and accepts $z \in (\Sigma^T_1)^w$. We show that there exist one composite service $CS$ over services in $C$ such that $CS|_{=} \varphi$. Since $A^\varphi_2$ is nonempty, there exists an accepting run $(T_r, r)$ of $A^\varphi_2$. Recall that $(T_r, r)$ is labeled with $(\Sigma^T_1)^* \times W \times cl(\varphi)$. By its definition, $A^\varphi_2$ corresponds to a computation tree $(T, V)$ of composite service $CS$ with start service $C_1$. We can construct a tree $(T_r, r')$ which is labeled with $T \times cl(\varphi)$ as follows.

1) $r'(c) = (c, s_{in})$.  
2) For every node $y \cdot c \in T_r$ with $r'(y) = (x, s)$ and $r(y \cdot c) = (z, w, s')$, we have $r'(y \cdot c) = (x \cdot (q, \sigma), s')$, where $q$ is a state of $C_i$, such that  
   a) if $q$ is a final state of $C_i$, we have $V(x \cdot (q, \sigma)) = L_i(w) \cup L_{f_i}(q_{\text{in}}(q))$; otherwise  
   b) $V(x \cdot (q, \sigma)) = L_i(w)$.  
As in the previous direction, it is easy to see that $(T_r, r')$ is an accepting run of $A^\varphi_2$ over $(T, V)$, which means $(T, V)$ satisfies $\varphi$. Thus $CS$ satisfies $\varphi$.

We show that the algorithm above is a single-exponential algorithm. Firstly, $A^\varphi_2$ is an alternating Büchi tree automaton with $O(2^n)$ states [22]. For every $A^\varphi_1$, by its definition, $A^\varphi_1$ is an alternating Büchi word automaton with $|W \times cl(\varphi)|$ states, which means that the number of states of $A^\varphi_1$ is polynomial on the sum of numbers of states in $C$ and $|\varphi|$. By [23], we can check the nonemptiness of $A^\varphi_1$ in time exponential in number of states of $A^\varphi_1$. So $CS(C, \varphi)$ can be solved in single-exponential time when $\varphi$ is a CTL formula.

**Lemma 2.** $CS(C, \varphi)$ is EXPTIME-hard when $\varphi$ is a CTL formula.
Proof. We verify the EXPTIME-hardness by reduction from the synthesis problem for CTL with complete information which is EXPTIME-complete [13]. The reduction is similar to the reduction in Theorem 2 of [15] except that LTL formula are replaced by CTL formula.

**Theorem 1.** \( \text{CS}(\mathcal{C}, \varphi) \) is EXPTIME-complete when \( \varphi \) is a CTL formula.

Proof. It follows from the upper bound and lower bound of \( \text{CS}(\mathcal{C}, \varphi) \) obtained with Lemma 1 and Lemma 2 that \( \text{CS}(\mathcal{C}, \varphi) \) is EXPTIME-complete when \( \varphi \) is a CTL formula.

**Theorem 2.** \( \text{CS}(\mathcal{C}, \varphi) \) is 2EXPTIME-complete when \( \varphi \) is a CTL* formula.

Proof. We first give a double-exponential algorithm for \( \text{CS}(\mathcal{C}, \varphi) \). Given \( \mathcal{C} \) and a CTL* formula \( \varphi \), the algorithm first constructs an alternating Rabin tree automaton \( A_\varphi \) which accepts exactly all the trees satisfying \( \varphi \), where \( A_\varphi \) has \( O(|\varphi|) \) states and \( O(|\varphi|) \) pairs (see [22] for details about \( A_\varphi \)). Then the algorithm constructs \( n \) alternating Rabin word automaton \( A^1_\varphi, \ldots, A^n_\varphi \) using a similar approach in Lemma 1, where every \( A_i^j \) (\( i \in \{1, n\} \)) accepts exactly all the trees that satisfy \( \varphi \) in which every tree is a computation tree of a composite service with start service \( C_i \) (\( i \in \{1, n\} \)). We can show in the same way in Lemma 1 that there exists a composite service satisfying \( \varphi \) over services in \( \mathcal{C} \) if and only if there exists at least one nonempty \( A_i^j \). By [24], an alternating Rabin word automaton \( A \) can be transformed into a nondeterministic Rabin word automaton \( A' \) with \( 2^{O(|\varphi|)} \) states and \( 2^{O(|\varphi|)} \) pairs, where \( p \) is the number of states of \( A \) and \( q \) is the number of pairs of \( A \). By [24], checking the emptiness of a nondeterministic Rabin word automaton can be down in time polynomial on the number of states and exponential in the number of pairs. So \( \text{CS}(\mathcal{C}, \varphi) \) can be solved in double-exponential time when \( \varphi \) is a CTL* formula.

Next we verify the 2EXPTIME-hardness by reduction from the synthesis problem for CTL* with complete information, which is 2EXPTIME-complete [13]. The reduction is similar to the reduction in Theorem 2 of [15] except that LTL formula is replaced by CTL*. Thus \( \text{CS}(\mathcal{C}, \varphi) \) is 2EXPTIME-complete when \( \varphi \) is a CTL* formula.

5 Environment-controllable synthesis of composite services

In this section, we investigate how to handle the failure in the synthesis of composite services. In \( \text{CS}(\mathcal{C}, \varphi) \), the synthesis fails if there exists no composite service over any part of services in \( \mathcal{C} \) which satisfies \( \varphi \). Fortunately, we find that we can cope with this situation by restricting the environment’s output behaviors. The idea is simple yet reasonable. Since component services proceed according to their business protocols, we can’t block their output messages when they receive messages from environment. But we can restrict the output behavior of environment (i.e. the input messages of services or users) such that the interaction sequences not satisfying \( \varphi \) are avoided. We illustrate this idea by an explanatory example.

**Example 1.** Consider a travel agent TA for planning a trip to Disney World including a round trip ticket and ticket for Disney World. There are two services FB and DR already in place for booking flight tickets and reserving tickets for Disney World, respectively.

Figure 1 shows the behavior protocols of FB (see Figure 1(a)) and DR (see Figure 1(b)). FB is a service intended to provide discounted flight tickets. Specifically, after receiving the request for booking flights (\(?fReq\)), it returns an offer if there are flights available (\(!fInfo\)); otherwise, it tells the user that there are no available flight tickets (\(!nTicket\)). For the first case, FB tells the user that the booking successes (\(!fPaid\)) after the user pays for the ticket (\(?fPay\)) or the booking fails (\(!fFailed\)) when the user rejects the offer (\(?fRej\)). According to its business protocol, FB don’t support refund of tickets since it only provides discounted tickets.

For DR, after receiving the request for reserving tickets for Disney World (\(?dReq\)), DR returns the price of tickets (\(?dPrice\)). DR provides users two options for payment. (1) It sells users discounted tickets which are nonrefundable (\(!dPaidDiscount\)) when receiving the payment immediately (\(?dPayDiscount\)). (2) Alternatively, it reserves full priced tickets for users (\(!dAdvPaided\)) when they pay in advance by credit card (\(?dAdvPay\)). In case of (2), when receiving the request for canceling the reservation (\(?dCancel\)), DR
tells users that the tickets are refunded (!dRefund). When the tickets are used, DR will receive the payment(!dPayFull) from users.

Since TA is intended to sell a trip to Disney World, we need to synthesize TA from FB and DR such that users can book both flight and tickets for Disney World or book none of them. In [16], we discussed this kind of correctness constraints which belong to safety properties and can be represented by CTL formula. Now we illustrate why and how we can restrict the outputs of users (i.e. the environment) to handle the violation of correctness constraints by TA.

Figure 1(c) shows a part of computation tree of TA where TA is synthesized from FB and DR through the interface functions represented as dotted lines in Figure 1. It is easy to see that the path composed of bold edges characterizes an undesirable scenario in which the user buys a nonrefundable flight ticket but canceled his ticket for Disney World. In order to avoid this scenario, the user should be informed that if he/she uses TA to book a trip to Disney, he can’t cancel the ticket for Disney World when he has bought the flight tickets, since he can’t get the money for flight tickets back. That is, the output of message of the user for canceling the ticket for Disney World is restricted.

In the rest of this section, we provide an formal investigation of the restriction to the environment’s output behaviors to cope with synthesis failure in CS(\(C, \varphi\)).

Recall that, in CS(\(C, \varphi\)), a composite service CS satisfies formula \(\varphi\) if and only if its computation tree \((T, \tau)\) satisfies \(\varphi\). Assume that CS enters state \(q_k\) after reading input sequence \(\sigma_k\) and CS can read input \(\sigma'_1, \ldots, \sigma'_i\) in state \(q_k\). At this moment, if the environment is not allowed to output \(\sigma'_i\), all the subtrees whose root is one of nodes in \(\{[\bar{\sigma}_k, q_k], ([\sigma_1, q], q) \in \delta(\bar{q}_k, \sigma'_i)\}\) should be pruned from the \((T, \tau)\), where \(\delta\) is the transition function of CS. Moreover, the environment cannot block the composite service; that is at least one of alphabet in \(\{\sigma'_1, \ldots, \sigma'_i\}\) should be provided by the environment. In this paper, we use the notion of execution tree in [25] to characterize how to prune nodes of computation tree. Intuitively, an execution tree is obtained from the computation tree \((T, \tau)\) by replacing the labels of nodes pruned in \((T, \tau)\) by \(\bot\). In doing so, the execution tree has the same shape with computation tree with difference only in their labeling.

Let \(C = (\Sigma^I, \Sigma^O, Q, q_{in}, \delta, \gamma, F)\). The computation tree \((T, \tau)\) of \(C\) is a \(\Sigma^O \cup \Sigma^I\)-labeled \(Q \times \Sigma^I\)-tree. Formally, an execution tree of \(C\) is a \(\Sigma^O \cup \Sigma^I \cup \{\bot\}\)-labeled \(Q \times \Sigma^I\)-tree, denoted by \((T, \tau')\), where

1) \(\tau'(\langle q_{in}, \varepsilon \rangle) = \tau(\langle q_{in}, \varepsilon \rangle)\).

2) For every node \([\bar{y}_k, \bar{\sigma}_k] \in T\), we have \(\tau'([\bar{y}_k, \bar{\sigma}_k]) = \tau([\bar{y}_k, \bar{\sigma}_k])\) if \(\tau'([\bar{y}_k, \bar{\sigma}_k]) \neq \bot\); otherwise, all the successors of \(([\bar{y}_k, \bar{\sigma}_k])\) are labeled with \(\bot\).

3) For each node \([\bar{y}_k, \bar{\sigma}_k] \in T\), if \(\tau'([\bar{y}_k, \bar{\sigma}_k]) \neq \bot\), there is at least one successor not labeled with \(\bot\).

**Problem statement.** In the rest of the section we shall study the following environment-controllable synthesis problems, referred to as LCS(\(C, \varphi\)). Given a library \(\mathcal{C}\) of component services and a temporal logic formula \(\varphi\), LCS(\(C, \varphi\)) is to determine whether or not there exists a composite service CS over (part of) services in \(\mathcal{C}\) such that there exists an execution tree of CS satisfying \(\varphi\).
We establish the complexity bounds of LCS(\mathcal{C}, \varphi) when \varphi is a CTL/CTL* formula.

**Lemma 3.** LCS(\mathcal{C}, \varphi) can be solved in single-exponential time when \varphi is CTL logic formula.

**Proof.** We develop an EXPTIME algorithm for LCS(\mathcal{C}, \varphi). Given \mathcal{C} and \varphi, the algorithm first constructs \(n + 1\) nondeterministic Büchi tree automaton \(N^\mathcal{C}_1(i \in [1, n])\) and \(N^\varphi\), where every \(N^\mathcal{C}_i(i \in [1, n])\) accepts exactly all the execution trees of all composite services with start service \(C_i(i \in [1, n])\), and \(N^\varphi\) accepts the set of trees that satisfy \(\varphi\). Thus there exists a composite service CS over \(\mathcal{C}\) such that there exists an execution tree of CS satisfying \(\varphi\) if and only if there exists a \(N^\mathcal{C}_i(i \in [1, n])\) such that the intersection of \(N^\mathcal{C}_i\) and \(N^\varphi\) is nonempty.

Given \(\mathcal{C} = \{C_1, \ldots, C_n\}\), where \(C_j = (\Sigma^I, \Sigma^O, Q_j, q^j_0, \delta_j, \gamma_j, F_j)(j \in [1, n])\), and \(\varphi\), formally, \(N^\mathcal{C}_1 = (\Sigma, D, Q, (q^1_{in}, \top), \delta, F)\), where

1. \(\Sigma = \Sigma^I \cup \Sigma^O \cup \{\bot\}\).
2. \(D = \bigcup_{i=0}^{n} \{d(q) \mid q \in Q_i\}\). That is, \(D\) contains all the branching degrees of services in \(\mathcal{C}\).
3. \(Q = \bigcup_{i=1}^{n} Q_i \times \{\top, \bot, \bot\}\) in \(N^\mathcal{C}_1\). Thus, each state \(q\) of every service in \(\mathcal{C}\) corresponds to three states \((q, \top), (q, \bot), (q, \bot)\) in \(N^\mathcal{C}_1\). Intuitively, when \(N^\mathcal{C}_1\) is in state \((q, \top)\), it can read only the letter \(\bot\). When \(N^\mathcal{C}_1\) is in state \((q, \bot)\), it can only read letters in \(\Sigma^I \cup \Sigma^O\). Finally, when \(N^\mathcal{C}_1\) is in state \((q, \bot)\), it can read both letters in \(\Sigma^I \cup \Sigma^O\) and \(\{\bot\}\).

4. \((q^1_{in}, \top)\) is an initial state.

5. The transition function \(\delta: Q \times \Sigma \times D \rightarrow 2^Q\) is defined as follows. Let \((q, b)\) be a state in \(Q_j \times \{\top, \bot, \bot\}\). When \(q\) is not a final state of \(C_j\), suppose that \(C_j\) can read input \(c_1, \ldots, c_i\) in state \(q\) and \(\text{succ}_{C_j}(q) = (q_1, q_2, \ldots, q_k)(k \geq l)\).

(a) For \(b \in \{\top, \bot\}\), we have \(\delta((q, b), b, \bot, k) = \{(q_1, \bot), (q_2, \bot), \ldots, (q_k, \bot)\}\).

(b) For \(b \in \{\top\}\), \(\delta((q, b), \gamma_j(q), k)\) is a collection of \(l\) \(k\)-tuple \(\alpha_1, \ldots, \alpha_k\), i.e. \(\delta((q, b), \gamma_j(q), k) = \{\alpha_1, \ldots, \alpha_k\}\).

(c) For \(b \in \{\top, \bot\}\), \(\delta((q, b), \bot, k)\) is a collection of \(n\) \(m\)-tuple \(\beta_1, \ldots, \beta_n\). (Recall that \(n = |\mathcal{C}|\)). Here \(\beta_i = (q^i_1, \bot, \ldots, q^i_{m}, \bot)\), where \(q^i_1, q^i_2, \ldots, q^i_{m}\) are all successors of initial state \(q^i_{in}\) of service \(C_i\).

(d) For \(b \in \{\top, \bot\}\), \(\delta((q, b), r_j(q), m)\) is a collection of \(n \cdot l\) \(m\)-tuples. Let \(\delta((q, b), r_j(q), m) = \{t_1, t_2, \ldots, t_n\}\) where \(t_i = \delta((q^i_{in}, b), r_i(q^i_{in}), m)\) is a set of \(l\) \(m\)-tuples which is computed in the same way in (b).

From the definition of acceptance of tree by a nondeterministic tree automaton, (a) and (c) above ensure that for every node \(v\) of trees accepted by \(N^\mathcal{C}\), if \(v\) is labeled with \(\bot\), all of its successors are labeled with \(\bot\); and (b) and (d) above prevent the environment from blocking the composite service. Obviously, \(N^\mathcal{C}_1\) accepts exactly all execution trees of composite services with start service \(C_i\). Let \(p\) be the maximal number of states of services in \(\mathcal{C}\). It is easy to see that \(|Q| \leq 3np\).

Recall that a node labeled with \(\bot\) of an execution tree stands for a node that actually does not exist. Accordingly, when interpreting CTL formulas over execution trees, we should treat a node labeled with \(\bot\) as a nonexistant node. To this end, we use the approach in [25] to define a function \(g(\psi)\) such that, for every CTL formula \(\psi\), \(g(\psi)\) restricts path quantification to paths that never visit a state labeled with \(\bot\). When \(\psi\) is a CTL formula, the formula \(g(\psi)\) may be not a CTL formula. But its path formulas have either a single linear-time operator or two linear-time operators connected by a Boolean operator [25]. By [26], such formulas have a linear translation to CTL. We don’t give the definition of \(g\) here due to the paper space limitation. Readers can refer to [25] for details.

So let \(N^\mathcal{C}\) be the Büchi tree automaton accepting exactly all the trees satisfying \(g(\varphi)\). As discussed above, there exists a composite service CS over services in \(\mathcal{C}\) such that there exist an execution tree of CS satisfying \(\varphi\) if and only if there exists at least one automaton in \(\{N^\mathcal{C}_i \mid i \in [1, n]\}\) such that the interaction with \(N^\varphi\) is nonempty. Equivalently, we have to test every \(N^\mathcal{C}_i \times N^\varphi (i \in [1, n])\) for emptiness. By [27], \(N^\varphi\) is a nondeterministic Büchi tree automaton and its number of states is exponential in \(|\varphi|\). So \(N^\mathcal{C}_i \times N^\varphi\) is also a nondeterministic Büchi tree automaton and its number of states is exponential in \(|\varphi|\) and
Lemma 4. LCS(\mathcal{C}, \varphi) is EXPTIME-hard when \varphi is a CTL formula.

Proof. We verify the EXPTIME-hardness by reduction from the satisfiability for CTL, which is EXPTIME-complete [28]. Given a CTL formula \varphi, we construct a library \mathcal{C} composed of only one service \mathcal{C}_1 and CTL formula \psi such that there exists a composite service \mathcal{C}_1 such that there exists a execution tree of CS satisfying \psi if and only if \varphi is satisfiable.

Assume that \varphi has \(l - 1\) existential quantifiers and \(m\) atomic propositions \(p_1, \ldots, p_m\). Let \(P = \{p_1, \ldots, p_m\}\). By the sufficient branching-degree property of CTL, \varphi is satisfiable if and only if there exists a \(2^P\)-labeled tree of branching degree \(l\) satisfying \varphi [29]. \(C_1 = (\Sigma^I, \Sigma^O, Q, q_0, \delta, \gamma, F)\) is defined as follows.

1) The state set \(Q = \{q_1, q_1', \ldots, q_m, q_m'\}\).
2) The initial state \(q_0 = q_1\).
3) The input alphabet \(\Sigma^I = \{b_1|l \in [1, m + 1]\} \cup \{\emptyset\}\).
4) The output alphabet \(\Sigma^O = P \cup \{e\}\), where \(e \notin P\).
5) The transition function \(\delta\) is defined as follows: for every \(q_i \in Q\), we have \(\delta(q_i, \emptyset) = q_j(i, j \in [1, l])\) and \(\delta(q_i, b_j) = q_i'(i \in [1, l], j \in [1, m + 1])\), \(\delta(q_i, \emptyset) = q_i(i \in [1, l])\), and \(\delta(q_i', \emptyset) = q_i'(j \in [1, m + 1])\).
6) The output function \(\gamma\) is defined as follows: for every \(q_i \in Q\), we have \(\gamma(q_i) = \{e\}(i \in [1, l])\), \(\gamma(q_i') = p_j, (j \in [1, m])\), and \(\gamma(q_m') = \emptyset\).
7) The set of final state \(F = \{q_2, q_3, \ldots, q_l\}\).

Intuitively, from the definition of \(C_1\), it follows that (a) every state \(q_i\) has \(q_i', \ldots, q_m, q_m'\) as successors; (b) \(C_1\) outputs \(p_i\) in state \(q'_i\) (\(i \in [1, m]\)), and outputs nothing in state \(q_m'\); and (c) \(C_1\) enters states \(q'_i\) only exactly after it reads \(b_j(i \in [1, m + 1])\) from the environment. Let \((T, V)\) be the computation tree of \(C_1\) and let \((T, \tau)\) be an execution tree of \(C_1\). Then we can obtain a tree \((T, \tau)\) with branching degree \(l\) from \((T, \tau)\) such that

1) \(T_1 \subseteq T\) and for every node \((\delta_k, \bar{q}_k) \in T_1\), all states appearing in \(\bar{q}_k\) only come from the set \(\{q_1, q_2, \ldots, q_l\}\).
2) For node \(v = v' \cdot (q_k, \sigma) \in T_1\), let all the successors of \(v\) be \(v_1, \ldots, v_k(k \leq m)\) which are all labeled with propositions in \(P\). Then \(\tau(v) = \bigcup_{i=1}^{k} \tau(v_i)\).

Intuitively, it is easy to see that \(v\) is labeled with \(p_i \in P\) if and only if there is one successor of \(v\) in \((T, \tau)\) labeled with \(p_i\). So we now have to define \(\psi\) such that whenever the formula \(\varphi\) refers to \(p_i\), the formula \(\psi\) will refer to \(\exists \gamma(p_i, \tau)\). In addition, the path quantification in \(\psi\) should be restricted to computations of \((T_1, \tau)\); that is, to paths that never meet \(\perp\). We obtain \(\psi\) from \(g(\varphi)\) by replacing each \(p_i\) in \(g(\varphi)\) by \(\exists \gamma(p_i, \tau)\) where the function \(g\) is used to restrict path quantification to paths that never visit a state labeled with \(\perp\).

Since the library \(C_1\) is only composed of one service \(C_1\), from the definition of composite service, there is only one composite service CS over \(C\) with the interface function \(f(q_0) = 1\) for each final state \(q_f\) of \(C_1\). Obviously, CS and \(C_1\) have the same computation tree. So we need only to show that \(\varphi\) is satisfiable if and only if there exists an execution tree \((T, \tau)\) of \(C_1\) satisfying \(\psi\). Assume that \(\varphi\) is satisfiable. According to the sufficient branching-degree property of CTL, there exists a labeling function \(\tau_1\) such that \((T_1, \tau_1)\) satisfies \(\varphi\). Then we can define an execution tree \((T, \tau)\) as follows: for a node \(v \in T_1\) and every successor \(v' \cdot (q_i', b_j)\) of \(v\) in \(T_1\), \(\tau(v' \cdot (q_i', b_j)) = \perp\) if \(p_i \notin \tau_1(v)\) (\(i \in [1, m]\)) and \(\tau(v' \cdot (q_m', b_{m+1})) = \perp\) if no \(p_i\) (\(i \in [1, m]\)) is in \(\tau_1(v)\). From the discussion above, it is easy to see that \((T, \tau)\) satisfies \(\psi\). Assuming now that there exists an execution tree \((T, \tau)\) of \(C_1\) satisfying \(\psi\), we can get the tree \((T_1, \tau_1)\) satisfying \(\varphi\) as discussed above. Obviously, by the sufficient branching-degree property of CTL, \(\varphi\) is satisfiable.

Theorem 3. LCS(\mathcal{C}, \varphi) is EXPTIME-complete when \varphi is a CTL formula.

Proof. It follows from the upper bound and lower bound of LCS(\mathcal{C}, \varphi) obtained with Lemma 3 and Lemma 4 that LCS(\mathcal{C}, \varphi) is EXPTIME-complete when \(\varphi\) is a CTL formula.
Theorem 4. LCS(\(C, \varphi\)) is 2EXPTIME-complete when \(\varphi\) is a CTL* formula.

Proof. We first give a double-exponential algorithm for LCS(\(C, \varphi\)). Given \(C\) and a CTL* formula \(\varphi\), the algorithm first constructs the same \(n\) nondeterministic Büchi tree automaton \(N_C(i \in [1, n])\) and \(N_\varphi\) as in Lemma 3, except that \(N_\varphi\) is a nondeterministic Rabin automaton where the number of states of \(N_\varphi\) is double-exponential in \(|\varphi|\) and the number of pairs of \(N_\varphi\) is exponential in \(O(|\varphi|)\) [30, 31]. By [11, 31], the nonemptiness problem of every \(N_C(i \in [1, n]) \times N_\varphi\) can be solved in double-exponential time in \(O(|\varphi|)\), which gives us an algorithm of double-exponential time complexity on \(|\varphi|\). Next we verify the 2EXPTIME-hardness by reduction from the satisfiability problem for CTL*, which is 2EXPTIME-complete [32]. The reduction is similar to the reduction in Lemma 4. Thus LCS(\(C, \varphi\)) is 2EXPTIME-complete when \(\varphi\) is a CTL* formula.

6 Conclusions

We have provided a formal treatment of the synthesis of composite service from branching logic CTL and CTL*, a problem about synthesizing a composite service from a given services library such that the composite service satisfies the correctness specified by CTL/CTL* formula. Furthermore, for the case of synthesis failure, we discuss the problem of how to restrict output behaviors of the environment to make synthesis successful. We establish the complexity bounds for these two problems and show that these two problems are both EXPTIME-complete and 2EXPTIME-complete for CTL and CTL*, respectively. There is much work to be done. We would like to develop efficient heuristic algorithms for synthesis. We also expect that practical PTIME cases can be identified in certain specific application domains.

Acknowledgements

This work was supported by National Basic Research Program of China (Grant No. 2011CB302602), National Nature Science Foundation of China (Grant No. 61103031), and Project of NLSDE (Grant Nos. SKLSDE-2010-ZX-01, SKLSDE-2010ZX-03).

References

10 Emerson E A, Clarke E M. Using branching time logic to synthesize synchronization skeletons. Sci Comput Program,
26 Kupferman O, Grumberg O. Buy one, get one free!!! J Logic Comput, 1996, 6: 523–539
Information for authors

SCIENCE CHINA Information Sciences (Sci China Inf Sci), cosponsored by the Chinese Academy of Sciences and the National Natural Science Foundation of China, and published by Science China Press, is committed to publishing high-quality, original results of both basic and applied research in all areas of information sciences, including computer science and technology; systems science, control science and engineering (published in Issues 1, 3, 5, 7, 9, 11); information and communication engineering; electronic science and technology (published in Issues 2, 4, 6, 8, 10, 12).


Papers published in Sci China Inf Sci include:

- REVIEW (20 printed pages on average) surveys representative results and important advances on well-identified topics, with analyses and insightful views on the states of the art and highlights on future research directions.
- RESEARCH PAPER (no more than 15 printed pages) presents new and original results and significant developments in all areas of information sciences for broad readership.
- BRIEF REPORT (no more than 4 printed pages) describes key ideas, methodologies, and results of latest developments in a timely manner.

Authors are recommended to use Science China’s online submission services. To submit a manuscript, please go to www.scichina.com, create an account to log in to JoMaSy (Journal Management System), and follow the instructions there to upload text and image/table files.

Authors are encouraged to submit such accompanying materials as short statements on the research background and area/subareas and significance of the work, and brief introductions to the first and corresponding authors including their mailing addresses with post codes, telephone numbers, fax numbers, and e-mail addresses. Authors may also suggest several qualified experts (with full names, affiliations, phone numbers, fax numbers, and e-mail addresses) as referees, and/or request the exclusion of some specific individuals from potential referees.

All submissions will be reviewed by referees selected by the editorial board. The decision of acceptance or rejection of a manuscript is made by the editorial board based on the referees’ reports. The entire review process may take 90 to 120 days, and the editorial office will inform the author of the decision as soon as the process is completed. If the editorial board fails to make a decision within 120 days, please contact the editorial office.

Authors should guarantee that their submitted manuscript has not been published before and has not been submitted elsewhere for print or electronic publication consideration. Submission of a manuscript is taken to imply that all the named authors are aware that they are listed as coauthors, and they have agreed on the submitted version of the paper. No change in the order of listed authors can be made without an agreement signed by all the authors.

Once a manuscript is accepted, the authors should send a copyright transfer form signed by all authors to Science China Press. Authors of one published paper will be presented one sample copy. If more sample copies or offprints are required, please contact the managing editor and pay the extra fee. The full text opens free to domestic readers at www.scichina.com, and is available to overseas readers at www.springerlink.com.

Subscription information

ISSN print edition: 1674-733X
ISSN electronic edition: 1869-1919
Volume 55 (12 issues) will appear in 2012

Subscription rates
For information on subscription rates please contact:
Customer Service
China: sales@scichina.org
North and South America:
journals-ny@springer.com

Outside North and South America:
subscriptions@springer.com

Orders and inquiries:
China
Science China Press
16 Donghuangchenggen North Street, Beijing 100717, China
Tel: +86 10 64015683
Fax: +86 10 64016350
Email: informatics@scichina.org
North and South America
Springer New York, Inc.
Journal Fulfillment, P.O. Box 2485
Secaucus, NJ 07096 USA
Tel: 1-800-SPRINGER or 1-201-348-4033
Fax: 1-201-348-4505
Email: journals-ny@springer.com

Outside North and South America:
Springer Distribution Center
Customer Service Journals
Haberstr. 7, 69126 Heidelberg, Germany
Tel: +49-6221-345-0, Fax: +49-6221-345-4229
Email: subscriptions@springer.com

Cancellations
must be received by September 30 to take effect at the end of the same year.

Changes of address: Allow for six weeks for all changes to become effective. All communications should include both old and new addresses (with postal codes) and should be accompanied by a mailing label from a recent issue. According to § 4 Sect. 3 of the German Postal Services Data Protection Regulations, if a subscriber’s address changes, the German Federal Post Office can inform the publisher of the new address even if the subscriber has not submitted a formal application for mail to be forwarded. Subscribers not in agreement with this procedure may send a written complaint to Customer Service Journals, Karin Tiks, within 14 days of publication of this issue.

Microform editions are available from: ProQuest. Further information available at http://www.il.proquest.com/uni

Electronic edition
An electronic version is available at springerlink.com.

Production
Science China Press
16 Donghuangchenggen North Street, Beijing 100717, China
Tel: +86 10 64015683 or +86 10 64034134
Fax: +86 10 64016350
Printed in the People’s Republic of China

Jointly published by
Science China Press and Springer