Volume 55 Number 3 March 2012 491–736 ISSN 1674-733X CN 11-5847/TP

中国科学：信息科学

SCIENCE CHINA
Information Sciences

Sponsored by
CHINESE ACADEMY OF SCIENCES
NATIONAL NATURAL SCIENCE FOUNDATION OF CHINA

info.scichina.com
www.springer.com/scp
www.springerlink.com

Special Focus on Information Fusion

Guest Editors: JING ZhongLiang, LEUNG Henry, LI YuanXiang

SCIENCE CHINA PRESS
Springer

Editorial Board

Supported by NSFC

SCIENCE CHINA
Information Sciences

Contents

Vol. 55 No. 3 March 2012

Special Focus on Information Fusion

RESEARCH PAPER

SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

March 2012 Vol. 55 No. 3: 638–649
doi: 10.1007/s11432-011-4460-z

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

Citation Deng T, Huai J P, Wo T Y. Complexity of synthesis of composite service with correctness guarantee. Sci China Inf Sci, 2012, 55: 638–649, doi: 10.1007/s11432-011-4460-z

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

*Corresponding author (email: dengting@act.buaa.edu.cn)
1) http://www.w3.org/TR/2007/REC-soap12-part0-20070427/
2) http://www.w3.org/TR/wsdl
3) http://www.ibm.com/developerworks/library/ws-bpel

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 ε 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* π 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 $x \cdot \pi$. Given an alphabet Σ, a Σ-labeled tree is a pair $\langle T, V\rangle$ where T is a tree and $V : T \to \Sigma$ maps each node of T to a letter in Σ.

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 $L : W \to 2^{\text{AP}}$ 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 $K = (AP, W, R, w_{\text{in}}, L)$ can be viewed as a tree $\langle T_K, V_K\rangle$ which is defined as follows.

(1) For he root ε of T_K, $V_K(\varepsilon) = w_{\text{in}}$.

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

A nondeterministic transducer is an nondeterministic finite automaton with outputs. Formally, a nondeterministic transducer is a tuple $M = (\Sigma^I, \Sigma^O, Q, q_{\text{in}}, \delta, \gamma, F)$ where Σ^I is a finite input alphabet, Σ^O is a finite output alphabet, Q is a finite set of states, $q_{\text{in}} \in Q$ is an initial state, $\delta : Q \times \Sigma^I \to 2^Q$ is a nondeterministic transition function, F is a set of final states, and $\gamma : Q \to \Sigma^O$ 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. $\text{succ}_M(q) = \{q' \in W | \exists \sigma \in \Sigma^I$, 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 $\langle q_{\text{in}}, \varepsilon\rangle$ such that $V_M(\langle q_{\text{in}}, \varepsilon\rangle) = \gamma(q_{\text{in}})$.

2) For any node $v \in T_M$ and $\langle q, \sigma\rangle \in Q \times \Sigma^I$, we have $V_M(v \cdot \langle q, \sigma\rangle) = \sigma \cup \gamma(q)$.

A transducer M satisfies a CTL/CTL* formula φ if and only if its computation tree $\langle T_M, V_M\rangle$ satisfies φ, denoted by $M \models \varphi$. For node $\langle q_{\text{in}}, \varepsilon\rangle \cdot \langle q_1, \sigma_1\rangle \cdots \langle q_k, \sigma_k\rangle \in T$, Let $\bar{q}_k = q_{\text{in}} q_1 \cdots q_k$ and $\bar{\sigma}_k = \varepsilon\sigma_1 \cdots \sigma_k$. We denote $\langle q_{\text{in}}, \varepsilon\rangle \cdot \langle q_1, \sigma_1\rangle \cdot \cdots \cdot \langle q_k, \sigma_k\rangle$ by $[\bar{q}_k, \bar{\sigma}_k]$.

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

1) The set of states $W = \{\langle q, a\rangle | \exists q'$ s.t. $q \in \delta(q', a)\} \cup \{\langle q_{\text{in}}, \varepsilon\rangle\}$.

2) The initial state $w_{\text{in}} = \langle q_{\text{in}}, \varepsilon\rangle$.

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 $R(\langle q', a'\rangle, \langle q, a\rangle)$ 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 $w = w_{\text{in}}$, 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 turple $A = (\Sigma, D, Q, q_{\text{in}}, F)$, where Σ is is a finite input alphabet, D is a set of directions, Q is a finite set of states, $q_{\text{in}} \in Q$ is an initial state, $\delta : Q \times \Sigma \times D \to 2^{D \times Q}$ 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 $\langle T, V\rangle$ is a $(T \times Q)-$tree $\langle T_r, r\rangle$ which satisfies the following:

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

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

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

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 $\wedge$ and $\vee$), where we also allow the formulas true and false and, as usual, $\wedge$ has precedence over $\vee$. For a subset $Y \subseteq X$ and a formula $\theta \in B^+(X)$, we say that Y satisfies θ if and only if assigning true to elements in Y and assigning false to elements in $X \setminus Y$ makes θ true. An alternating tree automaton is a tuple $A = \langle \Sigma, D, Q, q_{\text{in}}, \delta, F\rangle$ where Σ is is a finite input alphabet, D is a set of directions, Q is a finite set of states, $q_{\text{in}} \in Q$ is an initial state, $\delta : Q \times \Sigma \times D \to B^+(D \times Q)$ 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 $\langle T, V\rangle$ is a $(T \times Q)-$tree $\langle T_r, r\rangle$ and $\langle T_r, r\rangle$ satisfies the following:

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

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 θ, and for all $0 \leqslant i \leqslant d(x) - 1$, we have $y \cdot c_i \in T_r$ and $r(y \cdot c_i) = \langle x \cdot c_i, q_i\rangle$.

An alternating word automaton A is a tuple $A = \langle \Sigma, Q, q_{\text{in}}, \delta, F\rangle$ where Σ is a finite input alphabet, D is a set of directions, Q is a finite set of states, $q_{\text{in}} \in Q$ is an initial state, $\delta : Q \times \Sigma \to B^+(Q)$ 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 $a_0 a_1 \cdots$ is a $Q-$tree $\langle T_r, r\rangle$ and $\langle T_r, r\rangle$ satisfies the following:

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

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 $y_1, \ldots, y_k$ $(k \leqslant |Q|)$ such that $\{\gamma(y_1), \ldots, \gamma(y_k)\}$ satisfies θ.

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 nondeterministic transducer $C = (\Sigma^I, \Sigma^O, Q, q_{\text{in}}, \delta, \gamma, F)$ 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 Σ^I is the set of input messages and Σ^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 Σ^I and Σ^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 $C_i = (\Sigma^I, \Sigma^O, Q_i, q^i_{in}, \delta_i, \gamma_i, F_i)$ ($i \in [1,k]$). the interface function of C_i ($i \in [1,k]$) is defined as a function $f_i : F_i \to \{1, \ldots, k\}$ 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 $q_{\text{in}}^{f_i(q_f)}$ 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 nondeterministic transducer $\text{CS} = (\Sigma^I, \Sigma^O, Q, q_{\text{in}}, \delta, \gamma, F)$, where

1) The set of states is $Q = \bigcup_{i=1}^{n}(Q_i \times \{i\})$.

2) The initial state $q_{\text{in}} = \langle q^j_{\text{in}}, j\rangle$ and we refer to C_j as the start service of CS.

3) The transition function δ is defined as follows. For state $\langle q, i\rangle \in Q$, $\delta(\langle q, i\rangle, \sigma) = \{\langle q', i\rangle | q' \in \delta_i(q, \sigma)\}$ if $q \in Q_i \backslash F_i$ (*i.e.* q is not a final state of C_i); otherwise, $\delta(\langle q, i\rangle, \sigma) = \{\langle q', f_i(q)\rangle |\ q' \in \delta_{f_i(q)}(q_{\text{in}}^{f_i(q)}, \sigma)\}$.

4) The output function γ is defined as follows. For every state $\langle q, i\rangle \in Q$, $\gamma(\langle q, i\rangle) = \gamma_i(q)$ if $q \in Q_i \setminus F_i$; otherwise, $\gamma(\langle q, i\rangle) = \gamma_i(q) \cup \gamma_{f_i(q)}(q_{\text{in}}^{f_i(q)})$.

5) The set of final states is $F = \bigcup_{i=1}^{n}(F_i \times \{i\})$.

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

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

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

Proof. We develop an EXPTIME algorithm for $\text{CS}(\mathcal{C}, \varphi)$. Given $\mathcal{C} = \{C_1, \ldots, C_n\}$ and φ, the algorithm first constructs n alternating word automata A^i_φ ($i \in [1,n]$) such that every A^i_φ ($i \in [1,n]$) accepts exactly all the trees satisfying φ 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^i_φ such that (the languages of) A^i_φ is nonempty.

Given a CTL formula φ, we can construct in linear running time a Büchi alternating tree automaton $A_\varphi = \langle 2^{I \cup O}, cl(\varphi), s_{\text{in}}, \delta_\varphi, F_\varphi\rangle$ where $cl(\varphi)$ is the closure of φ (see [22] for detailed definition about closure) such that A_φ accept exactly the set of trees satisfying φ. As shown in [22], the product of A_φ and a Kripke structure K simulates a run of A_φ over the computation tree $\langle T_K, V_K\rangle$ of K. Intuitively, in order to construct A^i_φ ($i \in [1,n]$), 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_φ and K_i. We denote the product by A^i_φ.

2) Repeat the following process: for every state of A^i_φ which is not an initial state, if this state is associated with the Kripke structure K_j, compute the product of A^i_φ and K_j from this state. The process terminates and returns A^i_φ when there are no new states of A^i_φ.

Let $K_i = (\Sigma^I \cup \Sigma^O, W_i, w^i_{\text{in}}, R_i, L_i)$ $i \in [1,n]$. 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^i_φ which is defined as $A^i_\varphi = \langle \Sigma^i_\varphi, W \times cl(\varphi), \langle w^i_{\text{in}}, s_{\text{in}}\rangle, \delta^i_\varphi, F^i_\varphi\rangle$, 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) $\langle w^i_{\text{in}}, s_{\text{in}}\rangle$ is the initial state, where w^i_{in} and s_{in} are initial states of K_i and A_φ, respectively.

4) The transition function δ^i_φ 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 $\text{succ}_{K_i}(w) = \{w_0, \ldots, w_{d(x)-1}\}$ and $\delta_\varphi(s, L(w), d(w)) = \theta$. We have $\delta^i_\varphi(\langle w, s\rangle, b) = \theta'$, where θ' is obtained from θ by replacing each atom $\langle c, s'\rangle$ by the atom $\langle w_c, s'\rangle$.

b) otherwise, for every $j \in [1, n]$, let $\text{succ}_{K_l}(w^l_{\text{in}})=(w_0, \ldots, w_{d(w^l_{\text{in}})-1})$ (i.e. the set of all successors of initial state w^l_{in} of K_l) and $\delta_\varphi(s, L_j(w) \cup L_j(w^l_{\text{in}}), d(w^l_{\text{in}})) = \theta_l$ $(l \in [1, n])$. We have $\delta_\varphi(s, L_j(w) \cup L_j(w^l_{\text{in}}), d(w^l_{\text{in}})) = \theta'_l$, where θ'_l is obtained from θ_l by replacing each atom (c, s') in θ by the atom (w_c, s').

5) $F^i_\varphi = W \times F_\varphi$ is the acceptance condition of A^i_φ.

We show that there exists a composite service CS over services in library $\mathcal{C}$ such that CS$\models \varphi$ if and only if there exists at least one automaton in $\{A^1_\varphi, \ldots, A^n_\varphi\}$ which is nonempty.

Assume first that there exists a composite service CS over services $C_i (i \in [1, k])$ in $\mathcal{C}$ with interface functions $f_i (i \in [1, k])$ such that CS$\models \varphi$. Let the computation tree of CS be $\langle T, V\rangle$. Then we have $\langle T, V\rangle \models \varphi$. Recall that $\langle T, V\rangle$ is a $\Sigma^I \cup \Sigma^O$-labeled $Q \times \Sigma^I$-tree, where $Q = \bigcup_{i=1}^n (Q_i \times \{i\})$. Since A_φ accepts exactly all the trees satisfying φ, A_φ accepts $\langle T, V\rangle$. Thus there exists an accepting run $\langle T_r, r\rangle$ of A_φ over $\langle T, V\rangle$. 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 $\langle T_r, r'\rangle$ which is labeled with $(\Sigma^1_\varphi)^* \times W \times cl(\varphi)$ as follows. For every node $y \in T_r$ with $r(y) = \langle x, s\rangle$, 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 $x = x' \cdot \langle q, \sigma\rangle$. 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 $V(x) = L_i(w) \cup L_{f_i(q)}(q^{f_i(q)}_{\text{in}})$. From the definition of A^i_φ, it follows that there exists a sequence $z \in (\Sigma^1_\varphi)^*$ with length $|x|$ such that $r'(y) = \langle z, w, s\rangle$.

We show that $\langle T_r, r'\rangle$ is an accepting run of A^1_φ. In fact, since $F^1_\varphi = W \times F_\varphi$, we only need to show that $\langle T_r, r'\rangle$ is a run of A^1_φ; acceptance follows from the fact that $\langle T_r, r\rangle$ is accepting. Consider a node $y \in T_r$ with $r(y) = \langle x, s\rangle$. 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 $\langle T_r, r\rangle$ is a run of A_φ, 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 φ such that y has successors $y \cdot c_i$ and $r(y \cdot c_i) = \langle x \cdot c_i, s_i\rangle$ $(i \in [0, k-1])$. In $\langle T_r, r'\rangle$, by its definition, there exists $z \in (\Sigma^1_\varphi)^*$ with length $|y|$ and its k successor $z \cdot z_i$ where $z_i \in \Sigma^1_\varphi (i \in [0, k-1])$ such that $r'(y) = \langle z, w, s\rangle$ and $r'(y \cdot c_i)=\langle z \cdot z_i, w_{c_i}, s_i\rangle$ $(i \in [0, k-1])$. Let $\delta^i_\varphi(s, a) = \theta'$. By the definition of δ^i_φ, 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 θ'. When w is a final state of K_i, we can get the same result like above. Thus $\langle T_r, r'\rangle$ is a run of A^1_φ.

Assume *w.l.o.g* that A^1_φ is nonempty and accepts $z \in (\Sigma^i_\varphi)^w$. We show that there exist one composite service CS with start service C_1 over services in $\mathcal{C}$ such that CS satisfies φ. Since A^1_φ is nonempty, there exists an accepting run $\langle T_r, r\rangle$ of A^1_φ. Recall that $\langle T_r, r\rangle$ is labeled with $(\Sigma^i_\varphi)^* \times W \times cl(\varphi)$. By its definition, A^1_φ corresponds to a computation tree $\langle T, V\rangle$ of composite service CS with start service C_1. We can construct a tree $\langle T_r, r'\rangle$ which is labeled with $T \times cl(\varphi)$ as follows.

1) $r'(\varepsilon)=\langle \varepsilon, s_{\text{in}}\rangle$.

2) For every node $y \cdot c \in T_r$ with $r'(y) = \langle x, s\rangle$ and $r(y \cdot c)=\langle z, w, s'\rangle$, we have $r'(y \cdot c)=(x \cdot \langle q, \sigma\rangle, s')$, where q is a state of C_i, such that

(a) if q is a final state of C_i, we have $V(x \cdot \langle q, \sigma\rangle)=L_i(w) \cup L_{f_i(q)}(q^{f_i(q)}_{\text{in}})$; otherwise

(b) $V(x \cdot \langle q, \sigma\rangle) = L_i(w)$.

As in the previous direction, it is easy to see that $\langle T_r, r'\rangle$ is an accepting run of A_φ over $\langle T, V\rangle$, which means $\langle T, V\rangle$ satisfies φ. Thus CS satisfies φ.

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

Lemma 2. CS$(\mathcal{C}, \varphi)$ is EXPTIME-hard when φ 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. CS($\mathcal{C}, \varphi$) is EXPTIME-complete when φ is a CTL formula.

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

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

Proof. We first give a double-exponential algorithm for CS($\mathcal{C}, \varphi$). Given $\mathcal{C}$ and a CTL* formula φ, the algorithm first constructs an alternating Rabin tree automaton A_φ which accepts exactly all the trees satisfying φ, where A_φ has $2^{O|\varphi|}$ states and $O(|\varphi|)$ pairs (see [22] for details about A_φ). Then the algorithm constructs n alternating Rabin word automaton A^1_φ, ..., A^n_φ using a similar approach in Lemma 1, where every A^i_φ ($i \in [1, n]$) accepts exactly all the trees that satisfy φ 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 φ over services in $\mathcal{C}$ if and only if there exists at least one nonempty A^i_φ. By [24], an alternating Rabin word automaton A can be transformed into a nondeterministic Rabin word automaton A' with $2^{O(p)}$ states and $2^{O(q)}$ 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 CS($\mathcal{C}, \varphi$) can be solved in double-exponential time when φ 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 CS($\mathcal{C}, \varphi$) is 2EXPTIME-complete when φ is a CTL* formula.

5 Environment-controlable synthesis of composite services

In this section, we investigate how to handle the failure in the synthesis of composite services. In CS($\mathcal{C}, \varphi$), the synthesis fails if there exists no composite service over any part of services in $\mathcal{C}$ which satisfies φ. 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 φ 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 (!noTicket). 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

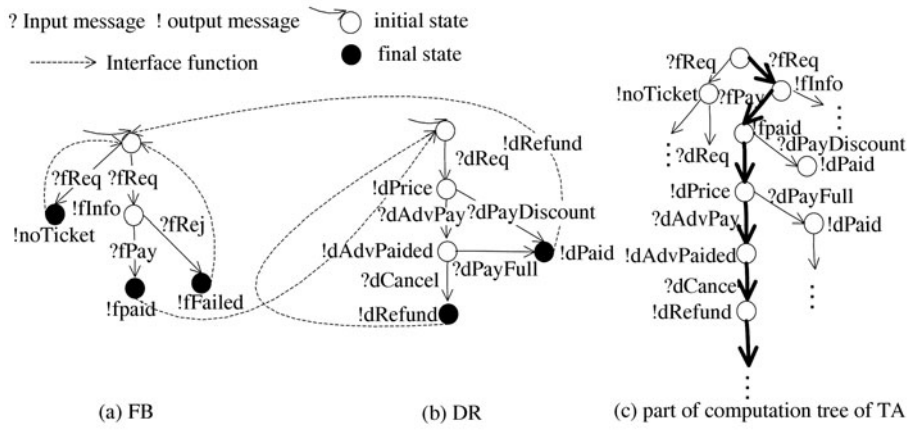


Figure 1 Synthesis of TA.

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 $\mathrm{CS}(\mathcal{C}, \varphi)$.

Recall that, in $\mathrm{CS}(\mathcal{C}, \varphi)$, a composite service CS satisfies formula φ if and only if its computation tree $\langle T, \tau\rangle$ satisfies φ. Assume that CS enters state q_k after reading input sequence $\bar{\sigma}_k$ and CS can read input $\sigma'_1, \ldots, \sigma'_l$ in state q_k. At this moment, if the environment is not allowed to output σ'_1, all the subtrees whose root is one of nodes in $\{[\bar{\sigma}_k, \bar{q}_k] \cdot \langle \sigma'_1, q\rangle | q \in \delta(q_k, \sigma'_1)\}$ should be pruned from the $\langle T, \tau\rangle$, where δ is the transition function of CS. Moreover, the environment cannot block the composite service; that is at least one of alphabet in $\{\sigma'_1, \ldots, \sigma'_l\}$ 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 $\langle T, \tau\rangle$ by replacing the labels of nodes pruned in $\langle T, \tau\rangle$ by $\perp$. In doing so, the execution tree has the same shape with computation tree with difference only in their labeling.

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

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

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

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

Problem statement. In the rest of the section we shall study the following environment-controllable synthesis problems, referred to as $\mathrm{LCS}(\mathcal{C}, \varphi)$. Given a library $\mathcal{C}$ of component services and a temporal logic formula φ, $\mathrm{LCS}(\mathcal{C}, \varphi)$ 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 φ.

We establish the complexity bounds of LCS($\mathcal{C}, \varphi$) when φ is a CTL/CTL* formula.

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

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

Given $\mathcal{C} = \{C_1, \ldots, C_n\}$, where $C_j = (\Sigma^I, \Sigma^O, Q_j, q_{\text{in}}^j, \delta_j, \gamma_j, F_j)(j \in [1,n])$, and φ, formally, $N_C^i = \langle \Sigma, D, Q, \langle q_{\text{in}}^i, \top\rangle, \delta, F\rangle$, where

1) $\Sigma = \Sigma^I \cup \Sigma^O \cup \{\perp\}$.

2) $D = \bigcup_{j=1}^n \{d(q) | q \in Q_j\}$. That is, D contains all the branching degrees of services in $\mathcal{C}$.

3) $Q = \bigcup_{j=1}^n Q_j \times \{T, \triangle, \perp\}$ in N_C^i. Thus, each state q of every service in $\mathcal{C}$ corresponds to three states $(q, \top)$, $(q, \triangle)$ and $(q, \perp)$ in N_C^i. Intuitively, when N_C^i is in state $(q, \perp)$, it can read only the letter $\perp$. When N_C^i is in state $(q, \top)$, it can only read letters in $\Sigma^I \cup \Sigma^O$. Finally, when N_C^i is in state $(q, \triangle)$, it can read both letters in $\Sigma^I \cup \Sigma^O$ and $\{\perp\}$.

4) $\langle q_{\text{in}}^1, \top\rangle$ is an initial state.

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

(a) For $b \in \{\triangle, \perp\}$, we have $\delta(\langle q, b\rangle, \perp, k) = \{\langle\langle q_1, \perp\rangle, \langle q_2, \perp\rangle, \ldots, \langle q_k, \perp\rangle\rangle\}$.

(b) For $b \in \{\top, \triangle\}$, $\delta(\langle q, b\rangle, \gamma_j(q), k)$ is a collection of l k-tuple $\alpha_1, \ldots, \alpha_l$, *i.e.* $\delta(\langle q, b\rangle, \gamma_j(q), k) = \{\alpha_1, \ldots, \alpha_l\}$. Here α_i $(i \in [1,l])$ is defined as follows: for every state $q' \in \delta_j(q, c_i)$, there is a state $\langle q', \top\rangle$ in α_i and for every state $q' \in \text{succ}_{C_j}(q) \backslash \delta_i(q, c_i)$, there is a state $\langle q', \perp\rangle$ in α_i.

When q is a final state of C_j, assume *w.l.o.g* that initial states of all services in $\mathcal{C}$ have the same branching degree m and every service in $\mathcal{C}$ can read the same number l of inputs.

(c) For $b \in \{\triangle, \perp\}$, $\delta(\langle q, b\rangle), \perp, m)$ is a collection of n m-tuples $\beta_1, \ldots, \beta_n$. (Recall that $n = |\mathcal{C}|$). Here $\beta_i = \langle\langle q_1^i, \perp\rangle, \ldots, \langle q_m^i, \perp\rangle\rangle$, where $\{q_1^i, q_2^i, \ldots, q_m^i\}$ are all successors of initial state q_{in}^i of service C_i.

(d) For $b \in \{\top, \perp\}$, $\delta(\langle q, b\rangle, r_j(q), m)$ is a collection of $n \cdot l$ m-tuples. Let $\delta(\langle q, b\rangle, r_j(q), m) = \{t_1, t_2, \ldots, t_n\}$, where $t_i = \delta(\langle q_{in}^i, b\rangle, r_i(q_{\text{in}}^i), m)$ is a set of l m-tuples which is computed 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_c^i, if v is labeled with $\perp$, all of its successors are labeled with $\perp$; and (b) and (d) above prevent the environment from blocking the composite service. Obviously, N_C^i 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 $|Q| \leqslant 3np$.

Recall that a node labeled with $\perp$ 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 $\perp$ as a nonexistent node. To this end, we use the approach in [25] to define a function g such that, for every CTL formula ψ, $g(\psi)$ restricts path quantification to paths that never visit a state label with $\perp$. When ψ 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_φ 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 φ if and only if there exists at least one automaton in $\{N_C^i | i \in [1,n]\}$ such that the interaction with N_φ is nonempty. Equivalently, we have to test every $N_C^i \times N_\varphi$ $(i \in [1,n])$ for emptiness. By [27], N_φ is a nondeterministic Büchi tree automaton and its number of states is exponential in $O(|\varphi|)$. So $N_C^i \times N_\varphi$ is also a nondeterministic Büchi tree automaton and its number of states is exponential in $O(|\varphi|)$ and

polynomial in the sum of numbers of states of services in $\mathcal{C}$. Also by [27], the nonemptiness problem of nondeterministic Büchi tree automata can be solved in quadratic time, which gives us an algorithm of exponential time complexity in $O(|\varphi|)$.

Lemma 4. LCS$(\mathcal{C}, \varphi)$ is EXPTIME-hard when φ 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 φ, we construct a library $\mathcal{C}$ composed of only one service C_1 and CTL formula ψ such that there exists a composite service CS over C_1 such that there exists a execution tree of CS satisfying ψ if and only if φ is satisfiable.

Assume that φ 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, φ is satisfiable if and only if there exists a 2^P-labeled tree of branching degree l satisfying φ [29]. $C_1 = (\Sigma^I, \Sigma^O, Q, q_{\text{in}}, \delta, \gamma, F)$ is defined as follows.

1) The state set $Q = \{q_1, \ldots, q_l, q'_1, \ldots, q'_m, q'_{m+1}\}$.

2) The initial state $q_{\text{in}} = q_1 \in Q$.

3) The input alphabet $\Sigma^I = \{b_l | l \in [1, m+1]\} \cup \{\emptyset\}$.

4) The output alphabet $\Sigma^O = P \cup \{e\}$, where $e \notin P$.

5) The transition function δ is defined as follows: for every $q_i \in Q$, we have $\delta(q_i, \emptyset) = q_j (i, j \in [1, l])$ and $\delta(q_i, b_j) = q'_j (i \in [1, l], j \in [1, m+1])$, $\delta(q_i, \emptyset) = q_i (i \in [1, l])$, and $\delta(q'_j, \emptyset) = q'_j (j \in [1, m+1])$.

6) The output function γ is defined as follows: for every $q_i \in Q$, we have $\gamma(q_i) = \{e\} (i \in [1, l])$, $\gamma(q'_j) = p_j$, $(j \in [1, m])$, and $\gamma(q'_{m+1}) = \emptyset$.

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 $q'_1, \ldots, q'_m, q'_{m+1}$ as successors; (b) C_1 outputs p_i in state q'_i $(i \in [1, m])$, and outputs nothing in state q'_{m+1}; and (c) C_1 enters states q'_i only exactly after it reads $b_i (i \in [1, m+1])$ from the environment. Let $\langle T, V \rangle$ be the computation tree of C_1 and let $\langle T, \tau \rangle$ be an execution tree of C_1. Then we can obtain a tree $\langle T_l, \tau_l \rangle$ with branching degree l from $\langle T, \tau \rangle$ such that

1) $T_l \subseteq T$ and for every node $\langle \bar{\sigma}_k, \bar{q}_k \rangle \in T_l$, all states appearing in $\bar{q}_k$ only come from the set $\{q_1, q_2, \ldots, q_l\}$.

2) For node $v = v' \cdot \langle q_i, \sigma \rangle \in T_l$, let all the successors of v be $v_1, \ldots, v_k (k \leqslant m)$ which are all labeled with propositions in P. Then $\tau_l(v) = \bigcup_{j=1}^{k} \tau(v_i)$.

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 $\langle T, \tau \rangle$ labeled with p_i. So We now have to define ψ such that whenever the formula φ refers to p_i, the formula ψ will refer to EXp_i. In addition, the path quantification in ψ should be restricted to computations of $\langle T_l, \tau_l \rangle$; that is, to paths that never meet $\perp$. We obtain ψ from $g(\varphi)$ by replacing each p_i in $g(\varphi)$ by EX$p_i (i \in [1, m])$, where the function g is used to restrict path quantification to paths that never visit a state labeled with $\perp$.

Since the library $\mathcal{C}$ is only composed of one service C_1, from the definition of composite service, there is only one composite service CS over $\mathcal{C}$ with the interface function $f(q_f) = 1$ 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 φ is satisfiable if and only if there exists an execution tree $\langle T, \tau \rangle$ of C_1 satisfying ψ. Assume that φ is satisfiable. According to the sufficient branching-degree property of CTL, there exists a labeling function τ'_l such that $\langle T_l, \tau'_l \rangle$ satisfies φ. Then we can define an execution tree $\langle T, \tau \rangle$ as follows: for a node $v \in T_l$ and every successor $v \cdot \langle q'_i, b_i \rangle$ of v in T, $\tau(v \cdot \langle q'_i, b_i \rangle) = \perp$ if $p_i \notin \tau'_l(v)$ $(i \in [1, m])$ and $\tau(v \cdot \langle q'_{m+1}, b_{m+1} \rangle) = \perp$ if no $p_i (i \in [1, m])$ is in $\tau'_l(v)$. From the discussion above, it is easy to see that $\langle T, \tau \rangle$ satisfies ψ. Assuming now that there exists an execution tree $\langle T, \tau \rangle$ of C_1 satisfying ψ, we can get the tree $\langle T_l, \tau_l \rangle$ satisfying φ as discussed above. Obviously, by the sufficient branching-degree property of CTL, φ is satisfiable.

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

Theorem 4. LCS($\mathcal{C}, \varphi$) is 2EXPTIME-complete when φ is a CTL* formula.

Proof. We first give a double-exponential algorithm for LCS($\mathcal{C}, \varphi$). Given $\mathcal{C}$ and a CTL* formula φ, the algorithm first constructs the same n nondeterministic Büchi tree automaton $N_C^i (i \in [1, n])$ and N_φ as in Lemma 3, except that N_φ is a nondeterministic Rabin automaton where the number of states of N_φ is double-exponential in $O|\varphi|$ and the number of pairs of N_φ is exponential in $O(|\varphi|)$ [30, 31]. By [11, 31], the nonemptiness problem of every $N_C^i \times N_\varphi$ $(i \in [1, n])$ 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($\mathcal{C}, \varphi$) is 2EXPTIME-complete when φ 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

1 Hull R, Su J W. Tools for composite web services: a short overview. SIGMOD Rec, 2005, 34: 86–95
2 Berardi D, Calvanese D, Giacomo G D, et al. Automatic composition of e-services that export their behavior. In: Orlowska M E, Weerawarana S, Papazoglou M P, et al., eds. Proceedings of International Conference on Service Oriented Computing. Trento: Springer, 2003. 43–58
3 Berardi D, Calvanese D, Giacomo G D, et al. Automatic composition of transition-based semantic web services with messaging. In: Bohm K, Jensen C S, Haas L M, et al., eds. Proceedings of International Conference on Very Large Data Bases. Trondheim: ACM, 2005. 613–624
4 Fan W, Geerts F, Gelade W, et al. Complexity and composition of synthesized web services. In: Lenzerini M, Lembothe D, eds. Proceedings of 27th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems. Vancouver BC: ACM, 2008. 231–240
5 Mitra S, Kumar R, Basu S. Automated choreographer synthesis for web services composition using i/o automata. In: Leymann F, Shan M C, eds. Proceeding of International Conference on Web Services. Salt Lake City: IEEE Computer Society, 2007. 364–371
6 Fu X, Bultan T, Su J. Realizability of conversation protocols with message contents. In: Zhang L J, ed. Proceeding of IEEE International Conference on Web Services(ICWS). San Diego: IEEE Computer Society, 2004. 96–103
7 Fu X, Bultan T, Su J. Analysis of interacting BPEL web services. In: Feldman S I, Uretsky M, Najork M, et al., eds. Proceedings of the 13th International Conference on World Wide Web. New York: ACM, 2004. 621–630
8 Deutsch A, Sui L, Vianu V, et al. Verification of communicating data-driven web services. In: Vansummeren S, ed. Proceedings of the 25th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems. Chicago: ACM, 2006. 90–99
9 Manna Z, Waldinger R. A deductive approach to program synthesis. ACM Trans Progr Lang Sys (TOPLAS), 1980, 2: 90–121
10 Emerson E A, Clarke E M. Using branching time logic to synthesize synchronization skeletons. Sci Comput Program,

1982, 2: 241–266
11 Pnueli A, Rosner R. On the synthesis of a reactive module. In: Proceedings of the 16th Annual ACM Symposium on Principles of Programming Languages. Austin: ACM, 1989. 179–190
12 Pnueli A, Rosner R. On the synthesis of an asynchronous reactive module. In: Ausiello G, Ciancaglini M D, Rocca S, eds. Proceeding 16th International Colloquium on Automata, Languages, and Programs. Stresa: Springer, 1989. 652–671
13 Kupferman O, Vardi M Y. Synthesis with incomplete information. In: Proceeding of 2nd International Conference on Temporal Logic. Manchester: Kluwer, 1997. 91–106
14 Finkbeiner B, Schewe S. Uniform distributed synthesis. In: Jagadeesan R, Jeffrey A, eds. Proceedings of 20th Annual Symposium on Logic in Computer Science. Chicago: IEEE Computer Society, 2005. 321–330
15 Lustig Y, Vardi M Y. Synthesis from component libraries. In: Alfaro L D, ed. Proceedings of 13th International Conference on Foundation of Software science and Computation Structures. New York: Springer, 2009. 395–409
16 Huai J P, Deng T, Li X X, et al. AutoSyn: A new approach to automated synthesis of composite web services with correctness guarantee. Sci China Ser F-Inf Sci, 2009, 52: 1534–1549
17 Pistore M, Traverso P, Bertoli P, et al. Automated synthesis of composite bpel4WS web services. In: Chang C K, Zhang L J, eds. Proceeding of International Conference on Web Services. Växjö: IEEE Computer Society, 2005. 293–301
18 Wang J S, Li Z J, Li M J. Composing semantic web services with description logics. J Softw, 2008, 19: 967–980
19 Qian Z Z, Lu S L, Xie L. Automatic composition of petri net based web services. Chinese J Comput, 2006, 29: 1057–1066
20 Pu K, Hristidis V , Koudas N. Syntactic rule based approach to Web service composition. In: Liu L, Reuter A, Whang K Y, et al., eds. Proceedings of International Conference on Data Engineering. Atlanta: IEEE Computer Society, 2006. 31–40
21 Boualem B F C, Toumani F. Representing, analysing and managing web service protocols. Data Knowl Eng, 2006, 58: 327–357
22 Orna K, Moshe Y V, Pierre W. An automata-theoretic approach to branching-time model checking. J ACM, 2000, 47: 312–360
23 Moshe Y V. Alternating automata and program verification. Comput Sci Today, 1995, 1000: 471–485
24 Muller D E, Schupp P E. Simulating alternating tree automata by nondeterministic automata: new results and new proofs of the theorems of Rabin, McNaughton and Safra. Theor Comput Sci, 1995, 141: 69–107
25 Kupferman O, Moshe Y V. Module checking. In: Alur R, Henzinger T A, eds. Proceedings of International Conference on Computer Aided Verification. New Brunswick: Springer, 1996. 75–86
26 Kupferman O, Grumberg O. Buy one, get one free!!! J Logic Comput, 1996, 6: 523–539
27 Vardi M Y, Wolper P. Automata-theoretic techniques for modal logics of programs. J Comput Syst Sci, 1986, 32: 183–221
28 Fischer M J, Ladner R E. Propositional dynamic logic of regular programs. J Comput Syst Sci, 1979, 18: 194–211
29 Emerson E A. Temporal and modal logic. In: Handbook of Theoretical Computer Science, Vol B. Cambridge: MIT Press, 1990. 997–1072
30 Vardi M Y, Stockmeyer L. Improved upper and lower bounds for modal logics of programs. In: Sedgewick R, ed. Proceedings of Annual ACM Symposium on Theory of Computing. Providence: ACM, 1985. 240–251
31 Emerson E A, Sistla A P. Deciding braching time logic. In: DemMillo R, ed. Proceedings of ACM symposium on Theory of computing. Washington DC: ACM, 1984. 14–24
32 Emerson E A, Jutla C S. The complexity of tree automata and logics of programs. SIAM J Comput, 2000, 29: 132–158

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).

Sci China Inf Sci is published monthly in both print and electronic forms. It is indexed by Science Citation Index, Current Contents, Computer and Information Systems Abstracts, Corrosion Abstracts, Electronics and Communications Abstracts, Engineered Materials Abstracts, CompuMath Citation Index, Solid State and Superconductivity Abstracts, Mathematical Reviews, CSA Mechan. Transport. Engineer. Abstracts, MathSciNet, Aluminum Industry Abstracts, BioEngineering Abstracts.

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 **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

SCIENCE CHINA PRESS

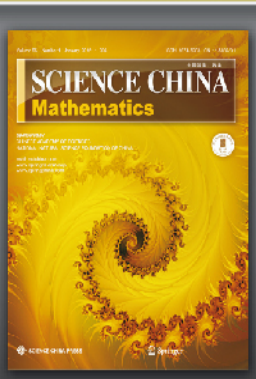
SCIENCE CHINA Series | Chinese Science Bulletin

SCIENCE CHINA Series and the *Chinese Science Bulletin* are academic journals supervised by the Chinese Academy of Sciences, co-sponsored by the Chinese Academy of Sciences and the National Natural Science Foundation of China, jointly published by Science China Press and Springer. *SCIENCE CHINA Series* and the *Chinese Science Bulletin* have presented the finest examples of China's development in both natural sciences and technological research to the international scientific community. In order to fully express China's achievements in fundamental scientific and engineering research, *SCIENCE CHINA Series* is published in seven journals, i.e., Mathematics, Chemistry, Life Sciences, Earth Sciences, Technological Sciences, Information Sciences, and Physics (including Mechanics and Astronomy), with the *Chinese Science Bulletin* serving as a multidisciplinary journal.

- **Peer-reviewed**
- **Indexed by SCI, CA, EI, etc.**
- **Online submission**
- **Easy access to the electronic version**

Honorary Editor General: ZHOU Guangzhao (Zhou Guang Zhao) | **Editor General:** ZHU Zuoyan

Mathematics

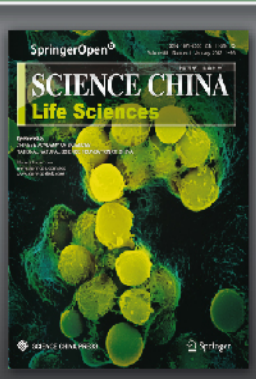


Monthly
Editor-in-Chief YANG Le (YANG Lo)

Chemistry

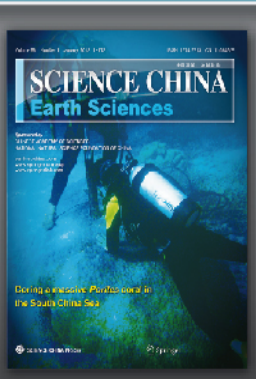
Monthly
Editor-in-Chief LI LeMin

Life Sciences



Monthly
Editor-in-Chief WANG DaCheng

Earth Sciences



Monthly
Editor-in-Chief SUN Shu

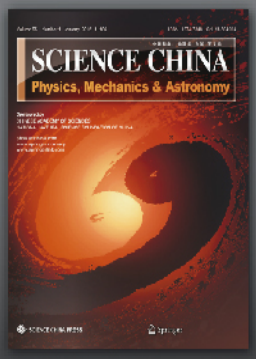
Technological Sciences

Monthly
Editor-in-Chief YAN LuGuang

Information Sciences

Monthly
Editor-in-Chief LI Wei

Physics, Mechanics & Astronomy



Monthly
Editor-in-Chief WANG DingSheng

Chinese Science Bulletin

Published three times every month
Editor-in-Chief XIA JianBai

www.scichina.com | www.springer.com/scp

Sponsored by
Chinese Academy of Sciences (CAS)
National Natural Science Foundation of China (NSFC)

Published by
Science China Press & Springer

ISSN 1674-733X