Journal of The Korea Society of Computer and Information
Vol. 24 No. 9, pp. 59-67, September 2019
www.ksci.re.kr
https://doi.org/10.9708/jksci.2019.24.09.059

Behavior-level Service Composition by Variable Abstraction

Hyun-Young Kil*

Abstract

The service composition based on Service-Oriented Architecture(SOA) can make us view various machines or its functionalities in the Web or Internet-of-Things environment as 'service', and efficiently create new value-added services that users want by compositing different services if there is no service to satisfy the client. The service composition problem with respect to behavioral descriptions deals with the automatic synthesis of a coordinator service that controls a set of services to reach a goal state. Despite its importance, however, solving the service composition problem with only partial observations remains to be doubly exponential in the number of variables in service descriptions, rendering any attempts to compute an exact solution for modest size impractical. Toward this challenge, in this paper, we propose novel approximation-based approaches using abstraction methods. We empirically validate that our proposals can solve realistic problems efficiently.

▸Keyword: Service Composition, Behavioral Description, Abstraction, Program Synthesis

I. Introduction

서비스 지향 아키텍처(Service-Oriented Architecture)의 서비스는 독립적 기능을 가질 수 있는 비즈니스 논리의 기본 단위다[1]. 서비스는 반복활용이 가능하고, 자체 독립적이며, 다른 서비스의 일부로 구성될 수 있다. 서비스의 내부 개발은 서비스의 소비자에게 "블랙박스"이나, 서비스 공급자는 해당 서비스를 이용할 수 있는 인터페이스(API)를 통해 서비스의 기능을 노출한다. 서비스 지향 아키텍처는 이러한 기능과 인터페이스를 조합하여 하나의 프로세스로 조율할 수 있고, 새로 형성된 프로세스는 새로운 기능을 제공하는 서비스가 된다.

서비스는 분산 시스템 환경, 웹 환경에서 서버시스템이 다수의 사람/기기 고객에게 제공하는 기능, 또는 자신의 기능을 사용할 수 있도록 지원하는 웹 서비스의 형태로 많이 사용되어 왔다. 이 뿐만 아니라, 최근 빠르게 발전하고 있는 사물인터넷(Internet-of-Things, IoT) 관련한 다수의 연구에서도 각각의 사물의 기능을 하나의 서비스로서 구현하고 상위 구조에서 이러한 서비스 집합을 관리하는 서비스 지향 아키텍처 기반 방식을 사용하고 있다[2, 3]. (그림 1)의 서비스 지향 아키텍처 기반 IoT 구조에서 IoT는 사물 추상화(Thing Abstraction) 계층, 서비스 관리(Service Management) 계층, 서비스 컴포지션(Service Composition) 계층, 응용프로그램(Application) 계층으로 구성된다. 이 구조를 기반으로 미들웨어나 운영체제는 IoT 기기들을 효율적으로 사용·관리할 수 있고 연계할 수도 있다. 본 연구의 서비스는 웹 서비스뿐만 아니라 IoT 기기에 대응하는 전반적 소프트웨어 서비스를 다 포함한다.

• First Author: Hyun-Young Kil, Corresponding Author: Hyun-Young Kil
*Hyun-Young Kil(hykil@kau.ac.kr), Dept. of Software, Korea Aerospace University
• Received: 2019. 05. 08, Revised: 2019. 06. 17, Accepted: 2019. 06. 17.
• This work was supported by 2019 Korea Aerospace University faculty research grant and by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (No. 2019R1F1A1062828).

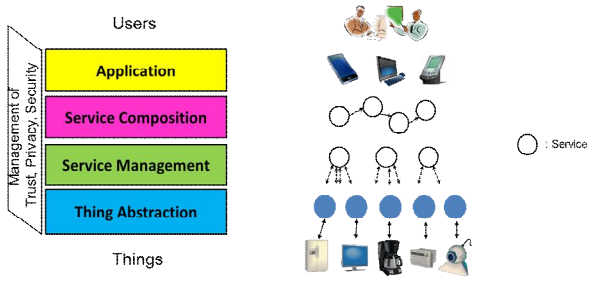


Fig. 1. SOA-based IoT-Service Architecture

행위 명세 기반 서비스 컴포지션은 행위 명세로 표현된 서비스들과 도달하고자 하는 목표가 주어졌을 때, 서비스들을 제어하여 목표에 도달하게 하는 조정자 서비스를 자동으로 합성하는 문제이다. 서비스의 "행위 명세"는 서비스가 사용자(사람이나 다른 서비스 등)와 상호작용을 하면서 내부적 또는 외부적으로 실행하는 활동에 대한 정형화된 명세로 정의된다.

다수의 서비스 컴포지션 연구들은 서비스의 모든 변수를 조정자 서비스가 알 수 있다고 가정한다. 이를 전체 관찰(Full Observation)이 가능한 환경이라 하며, 이 경우 조정자 서비스는 모든 서비스들의 정확한 상태를 파악하고 조정을 할 수 있다. 그러나, 실제 서비스들은 각각의 개발방식이 다르고 밖에서는 파악하기 어려운 내부 변수들을 갖고 있어, 외부에 있는 조정자 서비스는 변수의 일부만을 파악할 가능성이 높다. 이를 부분 관찰(Partial Oservation)이 가능한 환경이라 부르며, 이 경우 조정자 서비스는 서비스의 정확한 상태를 알 수 없으므로, 갖고 있는 변수값 정보를 기반으로 상태를 추정하여 서비스를 조정해야 한다. 부분 관찰(Partial Observation) 기반의 서비스 컴포지션이 실세계에서 일반적인 문제 유형이지만, 관련된 연구는 적다[4, 5]. 우리는 이전 연구에서, 합성된 조정자 서비스가 주어진 전체 서비스의 모든 변수들을 완전히 관찰할 수 있는 제한된 경우, 서비스 컴포지션 문제는 지수시간 계산 복잡도(EXP-hard)를 갖으며, 조정자 서비스가 모든 변수를 부분적으로 관찰하는 일반적 경우에 서비스 컴포지션 문제는 2의 지수 제곱시간 계산복잡도(2-EXP-hard)를 갖는다[6]. 이를 타개할 효율적 서비스 컴포지션 방안 연구로 우리는 추상화를 활용한 근사법 기반의 서비스 컴포지션 알고리즘을 제안한다.

첫 단계는 전체 서비스들이 원래보다 더 적은 수의 변수를 갖도록 추상화시키는 단계이다. 이 추상화 단계는 주어진 문제 내 서비스들의 가능한 모든 행동을 포함한 더 큰 행동모델(Over-approximate Behavioral Model)을 만들어 낸다. 이 과대평가된 서비스 행동모델을 제어하는 조정자 서비스를 만들 수 있다면, 조정자 서비스는 추상화 전의 서비스들도 제어하여 주어진 목표를 만족시킬 수 있다. 추상화된 행동모델에서 조정자 서비스를 만들 수 없다면, 변수를 추가하여 과대 평가된 서비스 행동모델을 더 구체화하고 다시 조정자 서비스를 만들기 위해 시도한다. 우리는 두 가지 추상화 방안을 제안하고, 이의 효과성을 보이기 위한 실험을 수행한다. 대표적 서비스 컴포지션 예제들에 대해, 추상화가 적용되지 않은 부분 관찰 기반 서비스 컴포지션 알고리즘[6]과 우리의 방식을 비교, 분석하였다. 2장에서는 관련 연구와 서비스 컴포지션 문제를 정의하고, 3장에서는 기본 알고리즘과 우리가 제안하는 알고리즘을 설명한 뒤, 실험결과를 보인다.

II. Preliminaries

1. Related work

자동화된 플래닝 기법은 문제의 유사성으로 인해, 서비스 컴포지션 문제에 많이 적용이 되어왔다. HTN(Hierarchical Task Network) 플래닝[7]은 집화된 행동의 표현을 허용한다. 플래너는 입력으로서 단일 단계의 동작들을 순서화시켜 연계하는 방법을 나타내는 복합 프로세스를 받는다. 그리고 최상위 레벨의 복합 프로세스의 실행 경로를 형성하는 원자 프로세스 사례 집합을 찾고자 한다. 그러나 이러한 연구들은 비결정론이나 부분 관찰에 대한 가능성을 고려하지 않는다.

어떤 연구들은 완전히 자동화된 컴포지션을 위한 서비스 프로세스 행위를 고려한다.[8, 9] 연구[8]는 웹 서비스에서 서비스 행위 표현에 대한 요구 사항과 함께, 이 행동 정보를 서비스 발견과 컴포지션을 위해 이용하는 이점에 대해 논의한다. 연구[9]는 경로 모델링, 분기 구조 식별 및 병렬 구조 파악 등을 사용하여 규칙의 오토마타에서 프로세스 모델을 합성한다.

서비스 컴포지션 문제는 게임 이론과도 밀접한 관련이 있다. 조정자는 특정한 웹 서비스의 비결정성을 제어할 수 없으며 서비스에 대한 입력만 결정해야 하므로, 이 문제는 부분 관찰이 존재하는 2인용 게임으로 간주될 수 있다. 그리고, 이 게임에서 조정자는 서비스들을 제어하여 원하는 결과에 도달하는 목적을 이루는 승리를 얻고자 한다. 연구[10]는 주어진 초기 위치에서 보편적인 게임의 결과를 결정하는 문제가 2의 지수제곱시간 계산복잡도(2-EXP-hard)라는 것을 증명했다. 연구[11]는 게임에서의 알고리즘 이론들과 이 이론들이 자동으로 프로그램을 합성하고 검증하는데 어떤 역할을 할 수 있는지 연구했다. 개방 시스템의 경우, 다양한 조정자 합성문제들이 시간 논리(Temporal Logic)를 기반으로 연구되었는데, 연구[12]는 부분 관찰 기반 환경에서 개방 시스템의 합성을 연구하였고, 계산 트리 구조(Computational Tree Logic)로 명세화된 문제가 지수시간-완전 계산 복잡도(EXP-complete)임을 증명했다.

서비스 컴포지션 문제는 부분 관찰을 통한 자동 플래닝기법과 긴밀한 관계가 있다. 연구[13]은 부분적인 관측 가능성을 고려한 계획을 위한 동적 논리방안을 제안했다. 부분 관측 가능성에 따른 플래닝의 복잡성은 연구[14]에서 연구되었고, 연구[15]는 시간적 추론에 기반하여 부분 관찰 가능성을 모델링하고 추론하는 방법을 모색했다. 연구[15]에서는 완전 자동 계획 도구인 모델 기반 플래너(Model Based Planner, MBP)가 추정 상태(Belief State)을 기반으로 개발되었다. 한편, 추상화를

사용한 플래닝 연구로, 연구[16]은 강화된 플래닝(Strong Plan)에 대한 관측 변수를 줄이기 위한 알고리즘을 제안했다. 그러나, 이 기술은 플랜이 수립될 때까지 변수를 식별할 수 없어 서비스 컴포지션 문제에 적용할 수 없다.

서비스 컴포지션문제에 대해 많은 연구[17, 18, 19, 20]가 수행되었지만 부분 관찰의 가능성을 갖는 현실적 모델을 사용하는 연구는 드물다. 연구[17, 18, 19]는 부분 관찰성을 가진 서비스 컴포지션을 정의하고 자동화된 플래닝 기법을 사용하여 알고리즘과 도구를 제시했다. 그러나 우리가 아는 한 서비스 컴포지션 문제에 대한 연구나 추상화를 사용한 부분 관찰에 대한 플래닝 알고리즘은 제안하지 않았다.

이전에는 웹 서비스들을 연계하는 연구가 다수였다면, 최근에 IoT 환경을 고려한 서비스 컴포지션 연구가 이루어지고 있다. IoT 미들웨어는 IoT 기기의 기능들을 각각의 서비스들로 고려, 개발함으로써 기존처럼 소프트웨어적 처리를 수행할 수 있다[3, 21, 22, 23, 24]. [23]은 이전의 안정적 서버 환경에 비해 실행시간 중에 서비스의 가용성이나 환경변화 등 IoT 환경의 특성을 고려한 동적 서비스 컴포지션을 연구하였다. [24]은 도시 컴퓨팅 환경에서 각 IoT 서비스를 사용자 관점에서 스마트 오브젝트로 구현하고, 사용자 작업을 달성하기 위해 주어진 일에 대한 IoT 서비스 컴포지션 방안을 연구하였다. 그러나, 아직 단순한 기능의 IoT 기기나 네트워크 관점에 초점을 맞춰 이루어지고 있다.

2. Service composition problem definition

우리는 서비스 w의 행위 명세를 (X, X^I, X^O, Init, T)이라 정의하며, 각 요소들을 아래와 같이 정의한다.

- X: w를 제어하는 변수의 유한집합으로, w의 상태(state) s는 X의 모든 변수 값들의 상태 집합이다. 우리는 모든 상태들을 포함하는 집합을 S라 한다.
- X^I: w가 읽어들이는 입력변수의 유한집합으로, $X \cap X^I = \emptyset$, 모든 변수를 포함하는 X와 X^I의 합집합은 유한한 범위(불, 제한된 정수, 또는 나열형 데이터종류)를 갖는다. 입력에 대한 상태 in은 X^I 내 모든 변수 값들의 상태집합이다. 우리는 모든 입력 상태들을 포함하는 집합을 S^I라 한다.
- $X^O \subseteq X$: w의 주변 환경에서 읽을 수 있는 출력변수의 유한집합이다. 입력변수와 출력변수의 집합을 X^{IO} ($X^{IO}= X^I \cup X^O$), 모든 변수의 집합을 X^A ($X^A=X \cup X^I$)라 한다.
- Init(X): X의 초기 술부 값으로, s가 초기상태이면 Init(s)는 true 값을 갖는다.
- T(X, X^I, X′): $X \cup X^I \cup X$의 전이술부으로, 변수 집합 X에 대해 $X'=\{x' \mid x \in X\}$은 다음 상태들을 나타내는 변수 집합을 의미한다. 상태 s에서 입력 in$\in S^I$을 받았을 때 s′가 다음 상태가 된다면, T(s, in, s′)는 true 값을 갖는다. T는 비결정형 전이관계 (non- deterministic transition relation)로 정의할 수 있다.

서비스 집합 W에 있는 모든 서비스들이 오직 조정자 서비스 c와 통신할 수 있다고 가정할 때, 서비스들의 집합인 W는 {w_1, …, w_n}, 각 서비스 w_i는 (X_i, X_i^I, X_i^O, $Init_i$, T_i)로 나타낼 수 있으며, X_i와 X_i^I 그리고 X_j와 X_j^I는 각각 서로소집합관계를 갖는다. W 역시 (X, X^I, X^O, Init, T)로 나타낼 수 있다.

조정자 c도 서비스이므로, c 역시 (X_c, X_c^I, X_c^O, $Init_c$, T_c)로 나타낸다. s_c는 조정자 서비스의 상태를, S_c는 조정자 서비스의 모든 상태들이 포함된 집합을 나타낸다. T_c는 원래 비결정형 전이관계로 정의되지만, 이 문제는 c에 대한 결정형 전이관계의 해결안이 더 이상적이다. 즉, 모든 조정자의 상태 s_c에서 입력값 in을 받았을 때 T(s_c. in, s_c')을 만족하는 s_c'가 오직 하나만 존재할 때, 조정자는 목적을 달성하는 정확하고 선명한 계획을 만들어낼 수 있기 때문이다. X에 대한 어떤 상태 s와 변수들의 집합 Y⊆X에 대해, s[Y]는 Y 내 변수들의 값의 집합을 나타낸다. 서비스의 집합 W=(X, X^I, X^O, Init, T)와 함께 $X^I=X_c^O$, $X^O=X_c^I$인 조정자 서비스 c=(X_c, X_c^I, X_c^O, $Init_c$, T_c)가 주어졌을 때, 우리는 서비스 집합 W에 대한 c의 실행트리(Execution Tree)를 "W ‖ c"로 표현하고 다음과 같이 정의할 수 있다.

- W ‖ c의 각 노드들은 $S \times S_c$에 있다.
- 루트 노드는 Init(s)=true, $Init_c(s_c)$=true인 (s, s_c)이다.
- 각 노드 (s, s_c)들은 다음을 만족하는 자식 노드들의 집합을 갖는다. {(s′, s_c') | T(s, in, s′)=true, in=$s_c[X^I]$, $T_c(s_c, in_c, s_c')$=true, in=s′[X^o]}.

직관적으로 서비스들의 집합인 W는 조정자 c의 현재 상태 s_c에서 입력값 in을 받아서 s에서 s′로 진행되고, 그 다음에 c는 서비스들의 새 상태인 s′로부터 입력값 inc을 받아서 s_c에서 s_c'로 진행한다. W와 c의 컴포지션은 동기화 통신의 형태이나, 우리는 r-전이을 사용하여 모델을 비동기화 통신 형태로 쉽게 확장할 수 있다. S의 부분집합인 G는 우리가 원하는 목적 상태들의 집합이며 술어(Predicate)으로서 명시된다. 서비스들의 집합 W, 조정자 서비스 c, 목적 G가 주어졌을 때, 첫 노드 (s^O, s_c^O)부터 목적 상태까지 도달하는 실행트리 W‖c의 모든 경로 (s^O, s_c^O)(s^1, s_c^1)…에 대해 $s_i \in$G이고 0보다 같거나 큰 i가 존재하면 우리는 "W ‖ c ⊧ G"으로 표현한다. 따라서, 서비스 컴포지션 문제는 주어진 서비스들의 집합 W와 목적 G에 대해, W ‖ c ⊧ G를 만족하는 조정자 서비스 c를 만드는 것이다.

또한, 서비스들의 모든 변수들의 값을 외부에서 관찰할 수 있는지 여부에 따라, 우리는 서비스 컴포지션 문제를 전체 관찰 기반의 서비스 컴포지션과 부분 관찰 기반의 서비스 컴포지션으로 나눈다. 전체 관찰 기반의 서비스 컴포지션은 서비스 컴포지션 문제의 특수한 경우로서, $X=X^O$ (즉, W가 내부변수를 갖고 있지 않는)을 만족하는 W(X, X^I, X^O, Init, T)라고 명시한다. 부분 관찰 기반의 서비스 컴포지션: 서비스 컴포지션 문제의 일

반적 경우로서, X^O에 대한 어떤 제약조건도 없다. 즉, 조정자 c는 X^O의 출력변수만을 읽을 수 있다.

전체 관찰 기반의 서비스 컴포지션 문제는 W의 변수 개수에 지수형태의 복잡도($O(2^n)$)를 갖는다. 이 복잡도는 다항식 공간의 한계를 갖는 교대 튜링 기계(Alternating Turing Machine, ATM) [26]를 활용하여 증명할 수 있다. 즉, 임의의 ATM A와 임의의 입력문자열 σ에 대해, 우리는 원하는 목적에 다다르는 조정자가 존재할 때만 A가 σ을 받아들이고, A가 σ을 받아들일 때에만 목적에 다다르는 조정자가 존재하는 서비스 컴포지션 문제를 다항식 시간(Polynomial time) 안에 만들 수 있다. 그러나, 부분 관찰 기반의 서비스 컴포지션 문제는 W의 변수 개수에 이중지수 형태의 계산 복잡도($O(2^{2^n})$)를 갖는다. 이 복잡도는 지수형태의 공간 한계를 갖는 ATM을 활용하여 증명할 수 있다.

[예제] 노약자를 위한 검진예약시스템

서비스 컴포지션 문제 정의에 대한 이해를 돕기 위해, 우리는 거동이 불편한 노약자를 위한 검진예약시스템을 사례로 들어 설명한다 (그림 2).

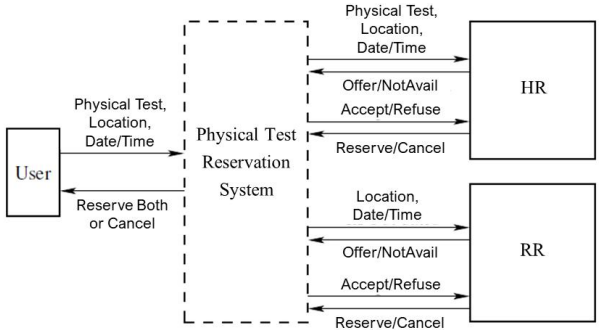


Fig. 2. Physical Test Reservation Systems

고객은 특정한 검진, 원하는 지역과 일정에 대응하는 병원과 차량을 예약하고자 한다. 역시 병원 예약 (HR) 서비스와 차량 예약 (RR) 서비스만 별도로 존재한다고 가정하였을 때, 기존에 존재하는 하나의 서비스만으로는 고객의 목적을 만족시킬 수 없다. 만족하는 서비스를 새로 개발하는 대신, 기존의 서비스를 결합, 사용함으로써 최종적으로 우리가 원하는 서비스를 구현, 사용하려 한다. 그 중 한 가지 방법은 병원 검진과 차량을 모두 예약하기 위해 서비스들과 통신하고 제어하는 조정자 서비스를 자동 구축하는 것이다. HR 서비스는 원하는 검진 종류, 지역, 날짜/시간을 포함하는 요청을 수신하고, 해당 시간 내 검진이 가능한 자리가 0보다 큰지 여부를 확인한다. 조건을 만족하는 자리가 존재한다면 병원 정보와 가격을 반환하고, 그렇지 않으면 "예약 불가(Not Available)"을 반환한다. 병원 정보와 가격을 제시한 후, 조정자는 사용자의 "동의" 또는 "거절"을 기다리고 대답에 따라 예약업무를 처리한다. 차량 (RR) 서비스 역시 출발/도착장소, 날짜/시간에 대하여 서비스가 가능한 차량 수를 확인하고, 사용 가능한 차량이 있는 경우에 차량 정보와 가격을 반환한다. 그렇지 않으면 "예약 불가(Not Available)"을 반환한다. 차량 정보와 가격을 제시한 경우, RR은 HR과 동일하게 해당 제안에 대한 사용자의 "수락" 또는 "거절" 응답을 받아 처리한다. 이 연구에서 우리는 사용자와의 상호작용도 포함하는 서비스들, 즉 좀더 복잡한 행동 프로세스를 갖는 서비스들을 먼저 고려한다. 따라서, 해당 연구는 이보다는 단순한 행동 프로세스를 갖는 IoT 서비스는 쉽게 포함, 확장할 수 있다.

위 예제의 서비스들이 고객의 응답, 즉 승인 또는 거절(각각 req1 또는 req2)을 통해 병원검진이나 차량을 예약 또는 거부할 수 있다고 가정할 때, 정의에 따라 서비스 w는 (X, X^I, X^O, Init, T)로 나타내며, 그 내용은 다음과 같다.

- X = {state, avail, reply, confirm, f_num, tr_num}이며, 변수 state은 {q1, q2} 중 하나의 값을, 변수 avail은 불타입의 값을, 변수 reply는 {undecided, offer, notAvail} 중에, 변수 confirm 값의 집합은 {undecided, reserve, cancel} 중 에, 변수 비행기 번호인 f_num은 {f1 , f2} 중에, 트랜잭션 번호를 의미하는 변수 tr_num은 {t1 , t2} 중 한개 값을 갖는다.

- X^I = {action}이며, 변수 action의 가능한 값의 집합은 {req1, req2, accept, refuse}이다.
- X^O = {reply, confirm, f_num}
- Init(X) ≡ (state = q1) ∧ (reply = undecided) ∧ (confirm = undecided).
- T(X. X^I, X′) ≡ (((state=q1) ∧ (action=req1) ∧ (avail =true))→((state′=q2) ∧ (reply′=offer) ∧ (tr_num′=t1))) ∧ (((state=q1) ∧ (action=req1))→(f_num′=f1)) ∧ … ∧ (((state=q2) ∧ (action=accept))→((state′=q1) ∧ (confirm′=reserve))) ∧ (((state=q2) ∧ (action=refuse))→((state′=q1) ∧ (confirm′=cancel))).

만약 변수가 갖을 수 있는 값의 범위가 유한하고 재귀형태가 없다면, WS-BPEL[27] 또는 OWL-S[28]와 같은 시멘틱 웹 기반 언어로 명세된 서비스의 프로세스 모델들은 원래 정보의 손실없이 위 정의 형태로 쉽게 변형, 표현될 수 있다.

앞의 예제는 병원 예약 (HR) 서비스와 차량 예약 (RR) 서비스만 별도로 존재한다고 가정한다. 또한, 고객은 병원 검진과 차량을 예약하려 한다. 그러므로, 서비스 집합 W와 최종 목표 G는 다음과 같이 정의할 수 있다.

- W = {w_{HR}, w_{RR}}
- G ≡ (PhysicaltestConfirm=reserve) ∧ (ridingConfirm=reserve)

III. The Proposed Scheme

1. AI planning based method

이 장에서는 앞의 문제 정의에 따라 일반적인 서비스 컴포지션, 즉 부분 관찰 기반의 서비스 컴포지션 문제를 푸는 기본 알고리즘을 설명한다. 연구[5]는 서비스 컴포지션 문제에 부분 관찰 기반의 플래닝기법을 적용하여 좋은 결과를 얻은 바 있기에, 이 연구에서 해당 알고리즘을 기본 알고리즘으로 활용한다.

일반적인 서비스 컴포지션 문제의 경우, 조정자 서비스는 전체 서비스들의 정확한 상태를 식별할 수 없다. 연구[5]는 이러한 불확실성을 추정 상태(Belief State)로 모델링하였다. 추정 상태는 현재 조건에서 목적 대상 서비스들의 가능하지만 확실히 판별은 할 수 없는 상태들의 집합으로, 기본 알고리즘은 초기 추정 상태에서 목표 추정 상태로 and-or 검색 트리를 구성해나가며 서비스 컴포지션을 위한 조정자를 구축한다. 해당 검색 트리는 서비스 출력 값의 비결정성으로 인해 임의의 추정 상태 노드에서 and edge들을 통해 자식 노드들의 집합을 갖도록 트리를 확장한다. 이 경우, 모든 자식 노드는 목표 추정 상태에 도달해야 한다. 조정자가 선택한 입력 값에 대해서는, or-edge를 통해 자식 노드들의 집합을 구성한다. 이 경우, 최소 하나의 자식노드는 목표 추정 상태에 도달해야만 한다.

and-or 검색 트리를 초기화하기 위해, 기본 알고리즘은 먼저 주어진 초기 술어, Init에 해당하는 루트 노드(추정 상태)를 만들고, “undecided"를 루트 노드의 결과값에 배정한다. 만약 Init에 해당하는 상태들이 이미 목적 상태에 포함이 된다면, 우리는 루트노드의 결과값에 "true"를 할당한다. 그 다음에는 (5-13행), 루트노드의 결과값이 결정될 때까지 다음 단계를 반복한다: (1) 아직 "true"와 "false"가 결정되지 않은 노드를 선택하고 (2) 선택된 노드로부터 가능한 다음에 이어진 노드들의 집합을 계산함으로써 트리를 확장하고 (3) 만약 노드가 and-or 제약조건에 따라 목적 상태에 도달하는지 여부를 체크한다. 각 노드의 결과를 확인하면 그 결과를 해당 위 노드에 전파한다. 결국 알고리즘이 루트 노드의 결과를 true로 식별하면, 실행트리로부터 조정자 서비스를 구성하고 반환한다. 그렇지 않다면, null 값을 반환한다. 알고리즘의 복잡도는 W의 변수 개수를 n이라 할 때, W의 상태 개수는 2^n이고 추정 상태는 2^{2^n}이므로, $O(2^{2^n})$이다.

```
- Input: a set of services and goal G
- Output: a coordinate service c
1  st := MakeSearchingTree(Init);
2  st.root.result := undecided;
3  if (States(Init) ⊆ States(G )) then
4       st.root.result := true;
5  while (st.root.result == undecided) do
6       node := SelectNode(st);
7       childNodes := ExtendST(st, node);
8       for childNode in childNodes
9            if (CheckSuccess(childNode)) then
10                node.result := true;
11           else if (CheckFailure(childNode)) then
12                node.result := false;
13      PropagateST(st, node);
14 if (st.root.result == true) then
15      return BuildCoordinator(st);
16 else return null;
```

Fig. 3. Basic Service Composition Algorithm

2. Internal variable abstraction based algorithm

서비스 컴포지션 문제는 계산 복잡도가 높은 어려운 문제(computationally-hard problem)으로 이를 해결하는 기본 알고리즘의 복잡도 역시 높다. 이에 대한 효율적 해결방법으로서, 우리는 추상화 기법을 사용하는 근사값 기반의 컴포지션 알고리즘을 제안하고자 한다.

먼저, 서비스들의 집합 W가 주어졌을 때 우리는 서비스 인터페이스를 통해 외부로 나타나는 I/O 변수들은 같으나 내부 변수는 추상화시킨 서비스들을 정의한다. 서비스 집합 W(X, X^I, X^O, Init, T)와 $X^{IO} \subseteq Y \subseteq X^A$인 변수 집합 Y가 주어지면, Y 기반의 내부 변수 추상화된 W는 다음 조건을 만족하는 $W_Y(X_Y, X^I_Y, X^O_Y, Init_Y, T_Y)$이다.

- $X_Y = Y \setminus X$, $X^I_Y = X^I$, $X^O_Y = X^O$.
- 모든 $s_Y \in S$에 대해, $(Init(s)=true) \wedge (s_Y = s[X_Y])$를 만족하는 $s \in S$가 존재한다면, $Init_Y(s_Y)=true$이며, 반대 방향 역시 성립한다.
- S에 속하는 모든 s_Y, s_Y'에 대해, $(T(s, in, s')=true \wedge (s_Y = s[X_Y]) \wedge (s_Y' = s'[X_Y])$를 만족하는 s, $s' \in S$가 존재한다면 $T_Y(s_Y, in, s_Y')=true$이며, 역방향도 성립한다.

W_Y는 W의 인터페이스를 그대로 갖고 있기 때문에, 위 정의에 기반한 W_Y에서 컴포지션이 될 수 있는 조정자 c가 만들어진다면, 역시 W에서도 c는 만들어질 수 있다. 더욱이, W_Y는 서비스 집합 W을 과대한 근사치로 만들었기 때문에 Soundness의 특성을 만족한다. 즉, 서비스들의 집합 W와 목적 G가 주어졌을 때, W을 signature-preserving 추상화한 W'에 대해 조정자 c가 $W' \| c \models G$를 만족한다면, c는 $W \| c \models G$를 만족한다.

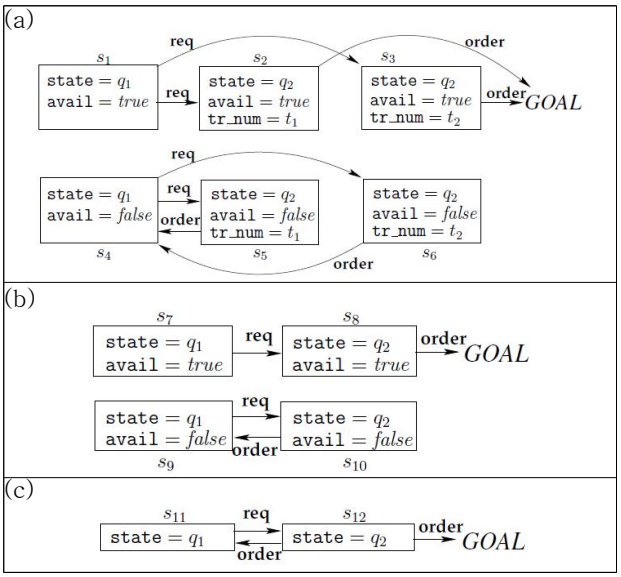


Fig. 4. (a)Original states (b)abstracted states (c) coarse abstracted states

그림 4(a)는 state, avail, tr_num의 세 가지 내부 변수가 있는 6개 상태를 가진 구체적인 상태 공간을 보여준다. 화살표 위의 기호는 입력 변수의 값을 나타낸다. 이 예에서, 우리는 req와 order를 시행하여 상태 s_1로부터 GOAL에 도달하는 것을 보장하는 계획을 만들어낼 수 있다. 그림 4(b)는 {state, avail}에 대한 추상적 상태 공간을 보여준다. 원래의 문제 공간에서의 s_1과 s_4는 각각 s_7과 s_9로 대응된다. 두 상태, s_2와 s_3, s_5와 s_6은 각각 s_8과 s_{10}으로 축소된다. 상태 수가 감소하더라도 원래 상태 공간의 모든 경로는 추상화된 공간의 경로 중 하나에 연계된다. 또한 s_1에 해당하는 s_7 상태로부터 우리는 여전히 GOAL에 도달할 수 있는 전략을 갖는다. 더 거친 추상을 보이는 그림 4(c)에서 s_1에 해당하는 s_{11}는 지나친 추상화로 인해 더 이상 GOAL에 도달 할 수 있는 계획을 갖지 못한다.

내부변수 추상화 기반 알고리즘의 핵심은 주어진 서비스 W를 W'로 추상화하고 W'에 대한 답인 조정자를 찾으려 한다는 것이다. 이러한 조정자를 찾을 수 있다면, 이는 원래 서비스 W를 제어하여 주어진 목표를 만족시킬 수 있다. 그렇지 않은 경우, 우리는 더 정밀한 추상화로 검색을 반복한다.

먼저, 우리는 입력변수와 출력변수만을 고려하여(즉, $Y := X^I \cup X^O$) W를 추상화한 W_Y을 구한다. 이 순간에 W_Y는 내부 변수를 포함하지 않으므로, 이 경우에 전체 관찰 기반의 서비스 컴포지션 알고리즘을 수행하며, 이 알고리즘은 기본 알고리즘보다 효율적인 EXP-hard 복잡도를 갖는다. 우리가 $W_Y \| c \models G$를 만족하는 조정자 c를 찾으면, 앞선 정의에 의해 c는 $W \| c \models G$를 만족한다. 그렇지 않다면, 더 많은 변수를 추가하여 현재의 W_Y를 정제하고 새로운 추상화에서 c를 찾고자 시도한다. 추가 변수 선택 방법은 다음 부분에서 설명한다. 우리는 $W_Y \| c \models G$를 만족하는 조정자 c 또는 추상화에 사용된 변수 집합이 원래 변수 집합과 같을 때까지 추상화/정제 과정을 반복한다. 후자의 경우는 주어진 문제에 대한 해결책이 없다는 것을 의미한다. 두 번째 반복문에서는 $O(2^{2^n})$ 복잡도를 갖는 부분 관찰 기반 서비스 컴포지션 알고리즘을 사용하지만, 일단 작은 추상화된 서비스 집합에서 조정자를 식별하면 변수 개수 면에서 검색 공간이 지수적으로 축소된다.

```
- Input: a set of services and goal G
- Output: a coordinate service c
1  Y := X^I ∪ X^O;
2  W_Y := MakeAbstractServices(W, Y);
3  if ((c := SC_FullObs(W_Y, G)) ≠ null) then
4      return c;
5  MakeDependencyGraph(W_Y, G);
6  while ((Var := SelectVars(W, G)) ≠ null) do
7      Y := Y ∪ Var;
8      W_Y := MakeAbstractServices(W, Y);
9      if ((c := SC_PartialObs(W_Y, G)) ≠ null) then
10         return c;
11 return null;
```

Fig. 5. Internal Variable Abstraction-based Service Composition Algorithm

만약 추상화된 서비스 집합에 대한 조정자 c를 찾지 못하는 경우에는 추상화가 너무 성글거나, 원래 문제에서 조정자가 존재하지 않기 때문이다. 후자의 경우, 최악의 경우가 원래 문제를 의미하므로, 알고리즘은 원래 문제에 해답이 없다고 올바르게 결론내릴 수 있다. 즉, 서비스들의 집합 W와 목적 G가 주어졌을 때, 만약 $W \| c \models G$를 만족하는 조정자 c가 존재하지 않는다면, 이 알고리즘은 최종적으로 null을 산출한다.

그러나 전자의 경우에는 원래 서비스 집합 W에 대한 조정자가 있음에도 불구하고, 추상화된 WY에 대해 null을 반환한 것이다. 그 이유는 목표에 도달하는데 중요한 정보가 포함된 변수를 포함하여 너무 많은 변수를 제거함으로써 지나치게 추상화되었기 때문이다. 이는 목표를 만족시키지 않는 상태들로 가는 실행 불가능한 경로를 유도한다. 예를 들어 그림 2(c)에서는 변수 avail이 없어짐으로써, s_1(s_2와 s_3)은 s4(s_5와 s_6)와 구별되지 않는다. 이는 순서에 따라 s_{12}에서 s_{11}까지의 실행 불가능한 경로를 만들게 되어, 더 이상 목표에 달성할 수 있는 전략이 없게 된다. 따라서 솔루션을 찾기 위해 더 많은 변수를 추가함으로써 현재의 추상화를 정제해야 한다. 목표를 만족시키지 못하는 경로는 적절한 조정자를 식별하지 못하기 때문에 주어진 목표 술어에 나타나는 변수의 값을 정확하게 추적하는 것이 중요하다. 이러한 이유로, 추가될 변수 선택의 가장 중요한 기준은 목표 술어 내부 변수와의 관련성이다. 목표 변수에 대한 각 변수의 관련성을 평가하기 위해 변수 종속성 그래프를 작성한다.

서비스들의 집합 W와 목적 G에 대해, 변수 종속성 그래프는 방향성 그래프 G(V, E)형태로 나타난다. 점의 집합인 V는 {x|x∈X∪XI}이며, 방향이 있는 선의 집합인 E는 {(x y)|x, y

∈V, y의 값은 x의 값에 의존한다}.

예를 들어 코드 "y := x"와 "if (x = true) then y := 0"은 y의 값이 x에 의존함을 의미한다. 앞선 예제의 주어진 W와 G에 대해 w_{AR}의 변수의 변수 종속 그래프의 일부를 생각해보자. 예를 들어 state, reply 및 tr_num의 값은 state, action 및 avail의 값에 따라 달라지므로(예제의 T 부분 참조) 우리는 이에 해당하는 방향선(state▷state), (action▷state), (avail▷state), …, (action▷tr_num)을 갖을 수 있다. 변수 의존성 그래프에서 목표 술어 안 변수에 대한 의존성이 강한 변수는 목표 변수에 더 가깝게 위치한다. 따라서, 알고리즘 1의 반복구조에서, SelectNewVars는 목표 술어 내 변수에 가장 가까운 홉을 갖는 변수의 세트를 반환한다(즉, 1-hop, 2-hop 등). 예를 들어, confirm은 목표 술어 내 변수이므로 1-hop 종속성을 갖는 변수집합은 {action, avail, state}이다.

3. Internal+α variable abstraction based algorithm

앞장에서는 추상화의 대상을 내부 변수로 제한했다. 즉, 추상화된 서비스는 원래 서비스와 동일한 입출력 변수를 갖는다. 그러나 대다수의 경우, 출력 변수 중 일부는 조정자의 결정에 중요한 정보를 제공하지 못한다. 예를 들어, 예제의 병원 검진 예약 서비스는 요청값(즉, req_1 및 req_2)을 병원예약슬롯 번호(즉, f_1 및 f_2)에 단순히 복사하고 그것을 참조용으로 고객에게 반환한다. 이 경우 조정자는 이 출력이 없어도 특정 서비스를 성공적으로 제어, 목표를 달성할 수 있다. 따라서, 이 절에서는 추상화의 대상으로 내부 변수뿐만 아니라 출력 변수도 고려하도록 한다.

먼저, 우리는 같은 입력변수를 갖고 보다 적은 수의 내부 변수와 출력변수를 갖는 추상화 서비스를 다음과 같이 정의한다. 서비스 집합 W(X, X^I, X^O, Init, T)와 $X^I \subseteq Y \subseteq X^A$인 변수 집합 Y가 주어지면, Y 기반의 내부변수+α 추상화된 W는 다음 조건을 만족하는 $W_Y(X_Y, X^I_Y, X^O_Y, Init_Y, T_Y)$이다.

- $X_Y = Y \setminus X^I$, $X^I_Y = X^I$, $X^O_Y = Y \cap X^O$.
- 모든 $s_Y \in S$에 대해, $(Init(s)=true) \wedge (s_Y = s[X_Y])$를 만족하는 $s \in S$가 존재한다면, $Init_Y(s_Y)=true$이며, 반대 방향 역시 성립한다.
- S에 속하는 모든 s_Y, s_Y'에 대해, $(T(s, in, s')=true \wedge (s_Y = s[X_Y]) \wedge (s_Y' = s'[X_Y])$를 만족하는 s, s′∈S가 존재한다면 $T_Y(s_Y, in, s_Y')=true$이며, 역방향도 성립한다.

내부변수와 출력변수를 추상화한 서비스 W_Y는 원래의 서비스 W보다 출력 변수가 적기 때문에, W_Y로 구성될 수 있는 조정자 c는 W의 불필요한 출력 변수를 무시함으로써(즉, $X^O \setminus X^O_Y$를 무시) W에서 구성될 수 있다. 게다가, W_Y는 W_Y의 모든 행동을 포함하고 있기 때문에 여기서 만들 수 있는 조정자는 원래 문제에서도 올바른 조정자의 역할을 할 수 있다.

추상화에 사용될 출력 변수를 선택하기 위해, 우린 앞장의 변수 종속성 그래프를 사용한다. 일반적으로 목표 술어 안 변수에 의존하는 내부 변수에 의존하는 출력 변수는 조정자가 서비스를 제어할 수 있도록 서비스 상태에 대한 중요한 정보를 제공하는 경향이 있다. 예를 들어, reply는 목표 변수 confirm에 대한 의존성이 있는 state과 avail에 의존성을 가지며 reply은 조정자가 사용 가능한 좌석이 있는지 여부를 추정하는 중요한 출력이다. 반면에 비행기 번호를 나타내는 f_num은 입력 변수, action에만 의존하며 조정자를 돕는 정보는 제공하지 않는다. 따라서 목표 술어 안의 변수에 의존성을 갖는 내부 변수에 의존하는 중요한 출력 변수의 집합 $X^{SO} \subseteq X^O$를 찾고, 그 다음 초기 추상화 단계에 X^{SO}를 사용한다. 즉, 이 추상화에서는 알고리즘 1의 첫 번째 행인 $Y := X^I \cup X^{SO}$를 시작한다. 나머지 출력 변수 (즉, $X^O \setminus X^{SO}$)는 마지막 반복단계에서 사용된다.

4. Experiment

우리는 모델 기반 플래너(MBP)[5]를 사용하여 내부변수의 추상화, 내부변수 이상의 추상화를 위한 자동화 툴을 구현했다. WS-BPEL로 명세된 서비스 명세 집합과 목표 술어가 주어지면, 목표를 달성하기 위해 주어진 서비스를 제어할 수 있는 조정자 서비스를 자동으로 구성한다. 우리의 툴이 조정자 서비스를 효율적으로 합성한다는 것을 입증하기 위해, 서비스 컴포지션 연구에서 널리 사용되는 서비스 예제 3세트(총 8가지)에 기본 알고리즘과 우리의 제안알고리즘을 적용하여 비교했다. 첫 번째 문제(P1)는 항공편과 숙박을 다 예약하는 여행에이젼트 시스템이다. 여기에는 3가지 경우가 있는데, 항공편 예약과 숙박 예약에 대한 입력 값으로 각각의 경우들은 4, 9 및 16개의 옵션을 갖고 있다. 두 번째 문제(P2)는 가구생산자서비스와 배송서비스라는 두 가지 서비스가 포함되어 있는 가구 구매시스템이다. 가구 생산자는 가구 품목을 생산하고 배송업체는 출발지에서 목적지까지 품목을 인도한다. 이 문제에도 세 가지 사례가 따로 있는데, 가구 주문 및 납품 주문에 대해 각각 4, 6 및 8개의 옵션이 있다. 세 번째 문제(P3)는 상점서비스와 은행서비스로 구성되어 있는 가상온라인상점(VOS)이다. 상점은 상품을 판매하고 은행은 한 계정에서 다른 계정으로 돈을 이체하는 기능을 갖고 있다. 여기에는 두 개의 사례가 포함되어 있는데, 품목 주문 및 송금 각각에 대해 3개, 4개의 옵션이 있다.

모든 실험은 2.4GHz 펜티엄 프로세서, 2GB 메모리, Linux 운영체제를 사용하는 PC에서 수행되었다. <Table 1>은 각 문제(Problem)에 대해 해당 문제에 존재하는 총 변수 개수(#Var)와 부울 값을 갖는 입/출력 변수 갯수(#I/Ovar)를 먼저 나타낸다. 그리고 해당 문제들을 AI 플래닝 기반 알고리즘(Basic)으로 문제를 해결했을 때의 시간과 이 연구에서 제안하는 방식인 내부변수 추상화 방식의 알고리즘(InternalVar Abstract)과 내부변수+α 추상화 방식의 알고리즘(InternalVar+α Abstract)의 문제 해결 시간을 각각 초 단위로 표시한다. 이 실험에서 7200초를 최대 수행시간으로 한정한다.

<Table 1>의 실험 결과에 따르면, 본 연구에서 제안하는 기

법들이 실행 시간 측면에서 기본(Basic) 알고리즘을 능가한다는 것을 보여준다. 일례로 문제 P1-1은 기본 알고리즘으로는 5.8초, 내부변수 추상화 방식의 알고리즘으로는 그 절반인 2.9초, 내부변수+α 추상화 방식의 알고리즘은 훨씬 작은 0.1초로 해결된다. P1-3, P2-3의 경우에는 기본 알고리즘과 내부변수 추상화 방식의 알고리즘으로는 7200초 안에서 해결되지 않지만, 내부변수+α 추상화 방식의 알고리즘으로는 실험시간 내에 해결된다. 문제별로 비즈니스 로직에서의 변수의 역할과 의존성에 따라 다른 차이가 있으나, 모든 추정 상태를 고려하여 and-or 검색 트리를 구성하는 기본 알고리즘은 검색 공간의 빠른 증가로 인해 효율성이 떨어지게 되기 때문이다. P2-2 외에 모든 문제가 기본 알고리즘, 내부변수 추상화 방식, 내부변수+α 추상화 방식으로 효율성이 높아지는 것을 보이며, 모든 문제에서 내부변수+α 추상화 방식의 알고리즘은 기본 알고리즘보다 훨씬 짧은 시간 안에 문제를 해결하는 효율성을 보였다.

본 실험은 중간 크기 정도의 예제를 사용했지만, 일반적으로 목표 변수와 관련있는 변수의 수가 제한적이므로 제안하는 추상화 기술은 더 큰 크기의 예제에도 유용하게 쓰일 수 있다. 예를 들어, 각각 10개의 변수가 있는 100개의 서비스(총 1,000개의 변수)에서 사용자가 원하는 목표는 종종 사용가능한 서비스 및 변수의 일부 (예 : 5%, 50개의 변수)와 관련된다. 이러한 경우, 우리의 기법은 95%의 무관한 변수를 제거하여 수렴 속도를 상당히 향상시킬 수 있다. 서비스 컴포지션 연구에서 일반적으로 WS-BPEL 또는 OWL-S의 동작 설명은 복잡하지 않아서 일반적으로 얕은 가변 종속성 그래프가 생성되기 때문에, 변수 의존성 그래프가 상대적으로 얕기 때문에 실험에서 반복 횟수는 2-3회 정도다.

Table 1. Experimental results

Problem	#Var	#I/Ovar	Basic	InternalVar Abstract	InternalVar+α Abstract
P1-1	38	9	5.8	2.9	0.1
P1-2	42	8	61.4	55.3	13.8
P1-3	69	10	-	-	162.0
P2-1	44	9	50.4	49.8	3.2
P2-2	55	10	320.0	364.6	42.3
P2-3	63	10	-	-	1214.0
P3-1	61	15	208.3	195.7	18.2
P3-2	74	15	3323.0	2321.3	520.8

IV. Conclusions

본 논문은 부분 관찰 환경에서 주어진 서비스 집합을 목표 상태에 도달하도록 제어하는 조정자 서비스를 자동으로 구성하는 근사법 기반의 서비스 컴포지션 기법을 제안했다. 제안된 솔루션은 우선 주어진 서비스의 행위가 원래보다 더 커진, 즉 관찰되는 변수가 적은 서비스들로 추상화한다. 그런 다음 추상화된 서비스들을 제어하여 목표에 도달할 수 있는 조정자 서비스를 구축하고자 한다. 조정자를 성공적으로 구축할 수 있다면, 이 조정자 서비스는 추상화 이전의 원래 주어진 서비스들을 제어하여 목표에 도달할 수 있다. 조정자를 찾을 수 없다면 중요한 변수를 우선적으로 추가하여 추상화모델을 정교화시킨다. 우리의 실험은 제안한 알고리즘의 효용성을 입증하는 결과를 나타냈다.

미래 연구를 위해 몇 가지 방향이 제시된다. 첫째, 결론을 조기에 수렴하기 위한 다른 추상화 방법을 연구할 필요가 있다. 둘째, 서비스 컴포지션 문제에 대한 우리의 기술을 보다 표현적인 목표(예 : 시간 논리로 지정)로 확장하고자 한다. 셋째, 문제를 해결하는 데 필요한 변수에 대한 엄격한 경계를 연구하고자 한다.

REFERENCES

[1] L. Papazoglou, M. Traverso, P. Dustdar, A. Schahram, and F. Leymann, “Service-Oriented Computing: State of the Art and Research Challenges,” IEEE Computer. Vol.40, pp.38-45, 2007.

[2] L. Atzori, A. Iera, and G. Morabito, "The internet of things: A survey," Computer Networks, Vol. 54, No. 15, pp. 2787-2805, 2010.

[3] V. Issarny, N. Georgantas, S. Hachem, A. Zarras, P. Vassiliadist, M. Autili, M. Gerosa, and A. Hamida, “Service-oriented middleware for the future internet: state of the art and research directions,” Journal of Internet Service and Application, Vol. 2, No. 1, pp. 23-45, 2011.

[4] F. Barbon, P. Traverso, M. Pistore, and M. Trainotti, "Run-time monitoring of instances and classes of web service compositions," Proceedings of International Conference on Web Services, pp. 63-71, 2006.

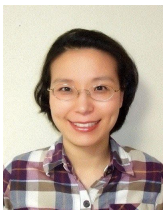
[5] P. Bertoli, A. Cimatti, M. Roveri, and P. Traverso, "Strong planning under partial observability," Artificial Intelligence, Vol.170, No.4, pp. 337-384, 2006.

[6] H. Kil, W. Nam, and D. Lee, "Computational complexity of web service composition based on behavioral descriptions," Proceedings of International Conference on Tools and Artificial Intelligence, pp. 359-363, 2008.

[7] G. Armano, G. Cherchi, and E. Vargiu, “A parametric hierarchical planner for experimenting abstraction techniques,” Proceedings of International Joint Conference of Artificial Intelligence, pp. 936-941, 2013.

[8] A. Brogi, “On the potential advantages of exploiting behavioural information for contract-based service discovery and composition,” Journal of Logic and Algebraic Programming, Vol. 80, No. 1, pp. 3-12, 2012.
[9] J. Yu, Y. Han, J. Han, Y. Jin, P. Falcarin, and M. Morisio. “Synthesizing service composition models on the basis of temporal business rules,” Journal of Computer Science and Technology, Vol. 23, No. 6, pp. 885-894, 2008.
[10] J. Rief, “The complexity of two-player games of incomplete information,” Journal on Computer and System Sciences, Vol. 29, pp. 274-301, 1984.
[11] W. Thomas. "Infinite games and verification," International Conference on Computer Aided Verification, Vol. 2404, pp. 58-64, 2002.
[12] O. Kupferman and M. Vardi, "Synthesis with incomplete information," Proceedings of International Conference on Temporal Logic, pp. 91-106, 1997.
[13] A. Herzig, J. Lang, D. Longin, and T. Polacsek, "A logic for planning under partial observability," Proceedings of National Conference on Artificial Intelligence, pp. 768-773, 2000.
[14] J. Rintanen, "Complexity of planning with partial observability," Proceedings of the International Conference on Automated Planning and Scheduling, pp. 345-354, 2004.
[15] M. Mott, "On the partial observability of temporal uncertainty," Proceedings of National Conference on Artificial Intelligence, pp. 1031-1037, 2007.
[16] W. Huang, Z. Wen, Y. Jiang, and L. Wu, "Observation reduction for strong plans," Proceedings of International Joint Conference of Artificial Intelligence, pp. 1930-1935, 2007.
[17] P. Traverso and M. Pistore, "Automated composition of semantic web services into executable processes," Proceedings of International Conference of Semantic Web, pp. 380-394, 2004.
[18] M. Pistore, P. Traverso, and P. Bertoli, "Automated composition of web services by planning in asynchronous domains," Proceedings of the International Conference on Automated Planning and Scheduling, pp. 2-11, 2005.
[19] Pistore et al., 2005b] M. Pistore, A. Marconi, P. Bertoli, and P. Traverso, "Automated composition of web services by planning at the knowledge level," Proceedings of International Joint Conference of Artificial Intelligence, pp.1252-1259, 2005.
[20] W. Nam, H. Kil, and D. Lee, "Type-aware web service composition using boolean satisfiability solver," Proceedings of IEEE International Conference on E-Commerce Technology and IEEE International Conference on Enterprise Computing, E-Commerce and E-Services, pp. 331-334, 2008.
[21] Autili M, Goldman A, Tivoli M, IEEE services visionary track on service composition for the future internet, IEEE World Congress on Services, pp. 327-328, 2015
[22] B. Cheng, M. Wang, S. Zhao, Z. Zhai, D. Zhu, and J. Chen, Situation-Aware Dynamic Service Coordination in an IoT Environment IEEE/ACM Transactions on Networking, Vol. 25, No. 4, pp. 2082-2095, 2017
[23] A. Bucchiarone, A. Marconi, M. Pistore and H. Raik, “A context-aware framework for dynamic composition of process fragments in the internet of services,” Journal of Internet Services and Applications, Vol. 8, No. 1, pp. 1-23, 2017
[24] I. Ko, H. Ko, A. Molina, and J. Kwon, “SoIoT: Toward a user-centric IoT-based service framework,” ACM Trans. Internet Technol., Vol. 16, No. 2, pp. 8-17, 2016.
[25] K. Görlach K, and F. Leymann, "A flexible engine for the unified execution of service compositions," IEEE Symposium on Service-Oriented System Engineering, pp 133-142, 2015
[26] Papadimitriou, and C. M. Papadimitriou, "Computational complexity," Addison-Wesley, 1994.
[27] WS-BPEL: Web services business process execution language version 2.0. http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.pdf, 2007
[28] OWL-S: Semantic Markup for Web Services https://www.w3.org/Submission/OWL-S/, 2004

Authors



Hyun-Young Kil received the B.S. and M.S. degrees in dept. of Computer from Korea University, Korea, in 1998 and 2001. She received the M.S degree in dept. of Computer & Info. Science from Univ. of Pennsylvania in 2003 and Ph.D in dept. of Computer Sci. & Engineering from the Pennsylvania State University, USA in 2010. Dr. Kil joined the faculty of the Department of Software at Korea Aerospace University, Goyang, Korea, in 2018. She is currently an assistant professor of dept. of Software, Korea Aerospace University. Her research interests include Web-based program synthesis, automated planning, and computer science education.