웹 서비스 코레오그라피를 이용한 자동 웹 서비스 컴포지션 시스템

이 상 규[†] · 한 상 용[††]

요 약

웹 서비스와 SOA(Service Oriented Architecture)의 확산으로 웹 서비스 컴포지션의 대한 관심도 점차 커져가고 있다. 최근에는 보다 동적이고 지능적인 SOA 환경의 구축을 위해서 자동 웹 서비스 컴포지션에 대한 연구가 활발하게 진행되고 있다. 하지만, 아직까지 자동 웹 서비스 컴포지션에 관한 완전한 해결책은 제시되지 못하고 있으며, 기존의 연구에서는 여러 가지 문제점을 드러내고 있다. 문법적인 정보 기반의 자동 컴포지션은 잘못된 의미 연결로 인하여 컴포지션의 정확성을 떨어뜨린다. 또한 대부분의 연구에서는 추상적인 컴포지션 결과를 만들어내기 때문에, 실제로 실행하기는 어려움이 있다. 본 논문에서는 이러한 문제점들을 개선하기 위하여 웹 서비스 코레오그라피 기반의 자동 웹 서비스 컴포지션 시스템을 제안하고 있다. 제안하는 시스템에서는 컴포지션의 정확성을 향상시켰고, 보다 구체적인 컴포지션 결과를 만들어낼 수 있다.

키워드 : 웹서비스 컴포지션, 코레오그라피, 자동 컴포지션

Automatic Web Services Composition System using Web Services Choreography

SangKyu Lee[†] · Sangyong Han[††]

ABSTRACT

Web Services composition has gained a considerable attention because of the widespread use of the Web Services and SOA. Recently, various researches on automatic Web Services composition are on going to realize more dynamic and intelligent SOA environments. However, there is no complete solution for automatic Web Services composition now and previous researches have several problems. Automatic composition based on syntactic information has low correctness through incorrect semantic linking. Moreover, many researches make an abstract process as the result of composition which is hard for actual execution. In this paper, improved automatic Web Services composition based on Web Services choreography is proposed. In this system, the correctness is improved and the result of composition is more concrete process.

Key Words : Web Service Composition, Choreography, Automatic Composition

1. 서 론

웹 서비스와 같은 SOA(Service Oriented Architecture) 환경에서는 기존에 존재하는 서비스들을 조합하여 새로운 애플리케이션을 만들 수 있다[1]. 이런 환경에서의 서비스 컴포지션은 수동으로 하는 컴포지션과 자동 컴포지션으로 나눌 수 있다. 수동 컴포지션은 조합할 서비스들의 수행 과정이나 조합 방법을 직접 정해주는 방식으로 WS-BPEL (Web Services Business Process Execution Language)[2]이 대표적인 예이다. 자동 컴포지션은 사용자가 요청을 하면, 컴포지션을 하는 시스템에 의해서 자동으로 컴포지션이 이루어지는 방식이다. 수동 컴포지션이 실제 산업계에서도 활용되고 있는 반면, 자동 컴포지션은 아직 연구단계에 머물고 있다.

본 논문에서는 자동 컴포지션의 정확성을 향상시키기 위하여 개선된 컴포지션 방법을 제안하고자 한다. 제안하는 방법은 기존의 문법적인 정보 기반의 방법을 개선한 것으로, 문법적인 정보 이외의 서비스들간의 연결 정보를 이용하여 자동 컴포지션의 성능을 향상시킨다. 이러한 연결 정보는 웹 서비스 코레오그라피(choreography)로부터 얻는다. 코레오그라피란 어떤 서비스가 다른 서비스들과 협력하여 보다 복잡한 형태의 작업을 수행할 수 있을 때 그러한 협력을 위한 구성을 의미한다. 제안하는 방법은 프로토타입 시스템으로 구현하였고, 기존 방법과의 비교를 통해 제안하는 방법을 평가하였다.

※이 논문은 2006년 중앙대학교 학술 연구비 지원에 의한 것임.
† 정 회 원 : 중앙대학교 대학원 컴퓨터공학과
†† 종신회원 : 중앙대학교 컴퓨터공학과 교수
논문접수 : 2007년 3월 9일, 심사완료 : 2008년 1월 3일

DOI: 10.3745/KIPSTD.2008.15-D.1.113

2. 관련연구

2.1 자동 웹 서비스 컴포지션에 관한 연구

웹 서비스 컴포지션은 웹 서비스 분야에서 가장 활발한 연구가 진행되고 있는 분야 중 하나로, 특히 자동 컴포지션과 관련된 연구가 주류를 이루고 있다. 자동 컴포지션의 연구는 크게 문법적(syntactic) 정보 기반의 컴포지션과, 의미적(semantic) 정보 기반의 컴포지션으로 나눌 수 있다.

문법적 정보 기반의 컴포지션에는 다음과 같은 연구들이 있어왔다. [3]에서는 매개자(mediator)가 전방향 추론(forward chaining) 방법을 통해서 자동화된 웹 서비스 컴포지션을 수행한다. 또한, AI 계획 수립(Artificial Intelligence planning) 알고리즘을 이용하여 자동 웹 서비스 컴포지션을 하기도 한다[4]. 이 밖에도 타입 유도 기법(type derivation)[5], 행위 네트워크(behavior network)[6]를 적용한 새로운 방법들도 제시되어 있다.

의미적 정보 기반 컴포지션 연구에서는 역방향 추론(backward chaining) 기법을 적용하거나[7, 8], 추론 네트워크(deduced network)[9], 인터페이스 오토마타(interface automata)[10]를 적용하기도 했다.

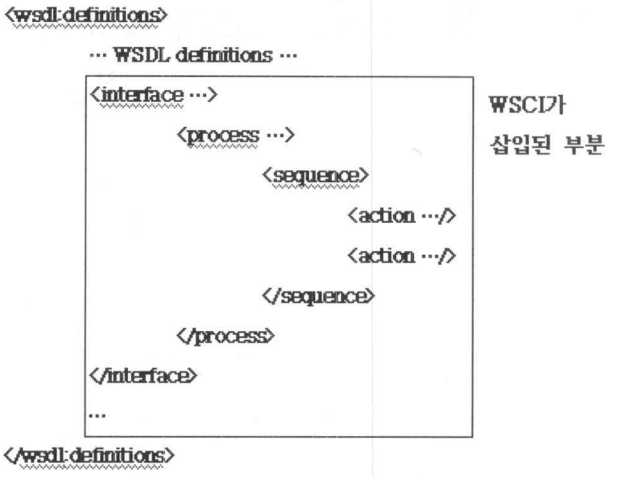
2.2 코레오그라피

웹 서비스에서 코레오그라피란 서비스들 간의 협력을 위한 구성을 의미하는 것이다. 이런 웹 서비스 코레오그라피를 기술하기 위해서 다양한 언어가 제안되었다. HP에서는 WSCL(Web Services Conversation Language)[9]을 제안하였고, BEA, Intalio, SAP 그리고 Sun Microsystems가 함께 WSCI(Web Services Choreography Interface)[11]를 제안하였다. Oracle에서는 WS-CDL(Web Services Choreography Description Language)[12]을 제안하였는데, 현재 WS-CDL이 W3C에서 채택되어 표준화가 진행되고 있다. 이들 언어의 목적은 웹 서비스가 다른 서비스들과 협력하기 위한 규칙을 표준화된 방법으로 기술하기 위한 것으로, 기존의 WSDL에서 기술하지 못했던 것을 보완하는 측면이 있다. WSDL이 웹 서비스의 정적인 인터페이스를 기술하는 것이라면, 이들 언어는 웹 서비스의 동적인 인터페이스를 기술해준다.

WSCL, WSCI, WS-CDL 등을 WS-BPEL과 같이 서비스 컴포지션을 위한 언어로 보는 경우도 있다. 하지만 이들 언어는 WS-BPEL과 같이 실행 가능한 컴포지션 결과를 만들어내지 못한다. 본 논문에서 컴포지션은 기존 서비스들의 조합을 통해 실행 가능한 새로운 서비스나 애플리케이션을 만들어내는 것으로 한정한다.

2.3 WSCI

본 논문에서는 자동 웹 서비스 컴포지션을 위해서 웹 서비스 코레오그라피를 이용하는데, 코레오그라피 기술 언어로 WSCI를 사용하였다. 현재 표준화가 진행되고 있는 언어는 WS-CDL이지만, 프로토타입 시스템의 구현상 편의를 위해 단순하면서도 풍부한 표현력을 제공하는 WSCI를 선택하였다. WSCI는 (그림 1)에서 보는 바와 같이 WSDL에 추가



(그림 1) WSDL에 삽입된 WSCI

적으로 코레오그라피를 기술하게 된다.

WSCI는 메시지 코레오그라피, 트랜잭션 처리, 예외 처리 등 다양한 부분에 대해서 기술할 수 있다. 이들 중 본 논문에서는 메시지 코레오그라피 부분을 이용하고 있다. 메시지 코레오그라피는 어떻게 메시지를 주고받으며 다른 서비스들과 협력을 할 수 있는지 나타내는 것이다.

3. 개선된 웹 서비스 컴포지션 시스템

3.1 자동 웹서비스 컴포지션 문제

웹 서비스는 주어진 입력에 대해서 출력을 만들어내는 함수로 볼 수 있다. 따라서 웹 서비스는 식 (1)과 같이 w_i라는 함수로 정의될 수 있다. IN_i은 입력의 집합을 나타내고, OUT_i은 출력의 집합을 나타낸다. 또, x와 y는 각각 입력과 출력의 파라미터로 사용된다. 실제 웹 서비스는 여러 개의 오퍼레이션으로 구성된다. 하지만 본 논문에서는 문제를 단순화시키기 위하여 하나의 서비스는 하나의 오퍼레이션으로만 구성된다고 가정하였다.

$$\begin{aligned} w_i(IN_i) &= OUT_i \\ IN_i &= \{x_{i,1}, x_{i,2}, \ldots\} \\ OUT_i &= \{y_{i,1}, y_{i,2}, \ldots\} \end{aligned} \quad (1)$$

웹 서비스는 레지스트리 역할을 하는 UDDI에 등록되어 관리된다. 또한 웹 서비스를 사용할 때는 이 UDDI를 검색하여 원하는 서비스를 찾게 된다. UDDI는 기존의 존재하는 웹 서비스들의 집합이라고 볼 수 있는데, N개의 서비스를 갖고 있는 레지스트리는 식 (2)와 같이 R로 정의할 수 있다.

$$R = \{w_1, w_2, \ldots, w_N\} \quad (2)$$

일반적으로 자동 웹 서비스 컴포지션에서는 사용자의 요청이 주어지면 그에 맞는 서비스 컴포지션 결과가 만들어진

〈표 1〉 서비스 수행 과정의 표기

종류	표기	설명
순차수행	$A \cdot B \cdot C$	A, B, C를 순차적으로 수행
병렬수행	$A \mid B \mid C$	A, B, C를 병렬로 수행
조건분기	$if(c_1, A) \cdot elif(c_2, B) \cdot else(C)$	조건 c_1을 만족하면 A 수행 조건 c_2를 만족하면 B 수행 그 외의 경우 C 수행
반복수행	$while(c_1, A)$ $until(B, c_2)$	조건 c_1을 만족할 때까지 A 수행 (조건 먼저 확인) 조건 c_2를 만족할 때까지 B 수행 (조건을 나중에 확인)

다. 또한, 사용자의 요청은 제공 가능한 입력의 집합과, 원하는 출력의 집합으로 이루어진다. 따라서 p 개의 입력이 제공 가능하고, q 개의 출력을 원하는 사용자의 요청은 식 (3)과 같이 *Request*로 정의될 수 있다.

$$\begin{aligned} Request &= \{IN_{req}, OUT_{req}\} \\ IN_{req} &= \{i_{req,1}, i_{req,2}, ..., i_{req,p}\} \\ OUT_{req} &= \{o_{req,1}, o_{req,2}, ..., o_{req,q}\} \end{aligned} \quad (3)$$

서비스 컴포지션의 결과는 여러 서비스들의 수행 과정으로 나오게 되는데, 순차적인 수행을 비롯하여, 반복수행, 조건에 따른 분기, 병렬적인 수행 등의 형태가 있을 수 있다. 이러한 수행 흐름의 표현은 웹 서비스 컴포지션에서 일반적으로 사용되는 표기법을 정리한 <표 1>을 본 논문에서 향후 사용하기로 한다. 예를 들어 w_1과 w_2가 순차적으로 수행되고, 조건 c_1에 따라서 w_3 또는 w_4를 수행하는 컴포지션 결과는 식 (4)와 같이 *Result*로 정의될 수 있다.

$$Result = w_1 \cdot w_2 \cdot (if(c_1, w_3) \cdot else(w_4)) \quad (4)$$

*Result*는 w_1, w_2 등의 함수가 순차수행, 조건분기의 규칙에 의해서 조합된 것이므로, *Result* 역시 식 (5)와 같은 하나의 함수로 볼 수 있다.

$$\begin{aligned} Result(IN_{res}) &= OUT_{res} \\ IN_{res} &= \{i_{res,1}, i_{res,2}, ..., i_{res,r}\} \\ OUT_{res} &= \{o_{res,1}, o_{res,2}, ..., o_{res,s}\} \end{aligned} \quad (5)$$

따라서 자동 웹 서비스 컴포지션 문제는 식 (2)와 같은 웹 서비스 레지스트리 R이 주어지고, 식 (3)과 같은 사용자의 요청 *Result*가 주어졌을 때, R에 있는 서비스들을 순차수행, 병렬수행, 조건분기, 반복수행의 규칙을 통해 조합하여 식 (6)의 조건을 만족시키는 식 (5)와 같은 함수 *Result*를 찾아내는 것이라고 정의할 수 있다.

$$\begin{aligned} IN_{res} &\subset IN_{req} \\ OUT_{res} &\supset OUT_{req} \end{aligned} \quad (6)$$

3.2 기존 연구의 문제점

기존의 연구들에서는 자동 컴포지션의 결과로 (그림 2)과 같은 서비스 수행의 흐름인 프로세스를 만들어낸다. S1, S2, S3, S4는 각각 개별적인 서비스를 나타내며, 서비스의 왼쪽에 기술된 것이 서비스의 입력을, 오른쪽에 기술된 것이 서비스의 출력을 나타낸다. A, B, C, D는 사용자로부터 주어진 입력이며, X와 Y는 사용자가 목표로 하는 출력이다. 각각의 수행 단계에서 필요로 하는 입력은 사용자로부터 주어진 입력이나, 이전의 서비스 수행의 출력에서 얻을 수 있다.

여기서 S1의 출력인 E와 F가 S2의 입력으로 쓰일 수 있다는 것은 S1과 S2의 WSDL 분석을 통해서 판단한 것이다. 둘 사이의 이름이 같기 때문에, 같은 것이라고 판단한 것이다. 하지만 단순히 문법적인 정보만를 이용하여 컴포지션을 수행할 경우, 그 결과의 정확성이 떨어지게 된다. 단순히 이름이 같다고 해서 같은 의미가 되는 것은 아니기 때문이다. 최근에는 시맨틱 웹 기술을 적용하여 이러한 단점을 보완한 방법들도 제안되었지만, 시맨틱 웹 자체가 아직은 연구단계에 있으며, 현재 사용되고 있는 웹 서비스 기술에는 적용되지 못하고 있다.

(그림 2) 자동 컴포지션 결과의 예

3.3 제안 시스템

(그림 3)는 제안하는 시스템의 아키텍처를 보여준다. 사용자의 요청이 주어지면 이것은 Translator에 의해서 내부적인 표현으로 변환된다. 변환된 것은 Composer로 전달되고, Composer는 자동 컴포지션 알고리즘을 통해 컴포지션을 수행한다. 이때 서비스들에 대한 정보와 코레오그라피 정보를 이용하게 되는데, 이들 정보는 Service Analyzer와 Choreography Analyzer를 통해서 얻을 수 있다. 컴포지션이 완료되면 그 결과는 다시 Translator로 전달되고, Translator는 이것을 변환하여 BPEL 프로세스를 생성한다. 사용자는 컴포지션 결과로 실행 가능한 BPEL 프로세스를 얻게 된다.

서비스 제공자는 서비스를 레지스트리에 등록해야 한다. 이때 서비스 레지스트리로는 UDDI를 그대로 사용한다. 본 논문에서는 WSCI를 사용하여 코레오그라피를 기술하는데, WSCI는 기존의 WSDL에 추가적으로 삽입되는 것이므로 UDDI는 그대로 사용 가능하다. Service Analyzer와 Choreography Analyzer는 UDDI에서 WSDL을 얻어와 분석하고, 서비스들에 대한 정보와 코레오그라피 정보를 준비한다. Service Analyzer와 Choreography Analyzer는 UDDI의 구독(subscription) 기능을 이용하는데, UDDI의 정보가 변경될 때마다 통지를 받아 업데이트를 한다.

3.3.1 Service Analyzer

Service Analyzer가 하는 일은 UDDI에 등록된 서비스들의 WSDL을 분석하여 3.1절에서와 같이 정형적인 형식으로 웹 서비스들의 정보를 준비하는 것이다. 이때 WSDL에서 자동 컴포지션에 필요한 정보인 <portType> 부분과 <message> 부분의 정보만을 사용하며, 바인딩 정보와 서비스 접근 주소를 갖고 있는 <binding>과 <service> 부분은 사용하지 않는다.

3.3.2 Choreography Analyzer

Choreography Analyzer는 WSDL에 삽입된 WSCI 부분을 분석하여 정형적인 형식으로 코레오그라피 정보를 준비하는 일을 한다. 코레오그라피 정보는 <표 1>의 방법으로 표현된다. 코레오그라피 정보는 두 가지 형태로 생성되게 된다. 하나는 WSCI에서 <process>에 해당하는 각각의 프로세스에 대하여 생성되는 정보이고, 다른 하나는 WSCI에서 <sequence>에 해당하는 각각의 순차적인 수행 흐름에 대하여 생성되는 정보이다. 물론 모든 <sequence>는 <process>에 내에 포함되어 있지만 자동 컴포지션 알고리즘에서 두 가지 정보를 따로 사용하게 되므로 두 가지 형태로 구분하는 것이다.

최종적으로 코레오그라피 정보는 식 (7)과 같이 모든 <process>에 대한 정보를 포함하는 집합인 CP 와, 모든 <sequence>에 대한 정보를 포함하는 CS 로 생성되게 된다.

$$CP = \{p_1, p_2, ..., p_L\}$$
$$CS = \{s_1, s_2, ..., s_M\} \quad (7)$$

3.3.3 자동 컴포지션 알고리즘

본 논문에서는 자동 컴포지션을 위해서 역방향 추론 방법을 기반으로 3.2절에 언급했던 기존의 문제점을 개선한 알고리즘을 제안한다.

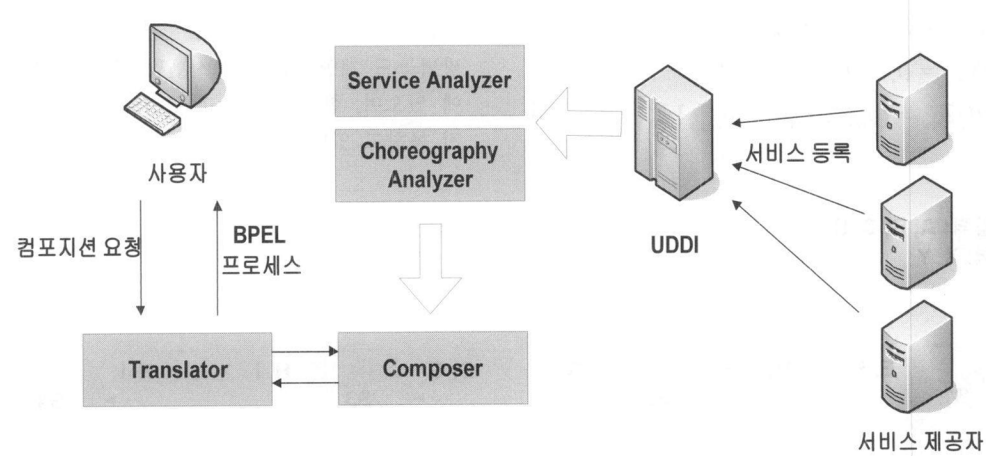
웹 서비스는 주어진 입력에 대해서 정해진 타입의 출력을 만들어낸다. 그러므로 식 (1)에서 정의했던 하나의 웹 서비스는 식 (8)과 같은 하나의 규칙으로도 볼 수 있다. 또한 식 (3)에서 정의했던 사용자의 요청 역시 식 (9)와 같은 하나의 규칙으로 볼 수 있다.

$$x_{i,1} \wedge x_{i,2} \wedge x_{i,3} \wedge ... \rightarrow y_{i,1} \wedge y_{i,2} \wedge ... \quad (8)$$

$$i_{req,1} \wedge i_{req,2} \wedge ... \wedge i_{req,p} \rightarrow o_{req,1} \wedge o_{req,2} \wedge ... \wedge o_{req,q} \quad (9)$$

따라서 기존에 존재하는 서비스들과, 사용자의 요청이 주어지게 되면, 정방향 추론 또는 역방향 추론 방법을 통해 사용자의 요청을 만족시키는 서비스 체인(chain)을 만들어낼 수 있다. 이러한 체인을 만들어내는 과정이 곧 서비스 컴포지션이 되며, 만들어진 체인이 바로 서비스 컴포지션의 결과가 되는 것이다.

역방향 추론 방법 기반의 알고리즘에서는 현재의 서비스 체인 이전에 연결될 서비스를 찾는 과정이 있다. 기존의 알



(그림 3) 제안시스템 아키텍처

고리즘에서는 이전 서비스를 찾기 위해서 단순히 레지스트리 검색을 통해 필요로 하는 출력을 제공해주는 서비스를 찾는다. (그림 4)은 이러한 과정을 그림으로 나타낸 것이다.

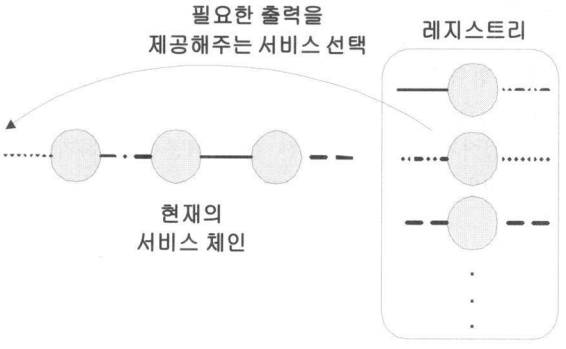
하지만 본 논문에서 제안하는 알고리즘에서는 다음의 세 단계로 이전에 연결될 서비스 또는 서비스 체인을 검색한다.

3.3.3.1 WSCI의 <process> 검색

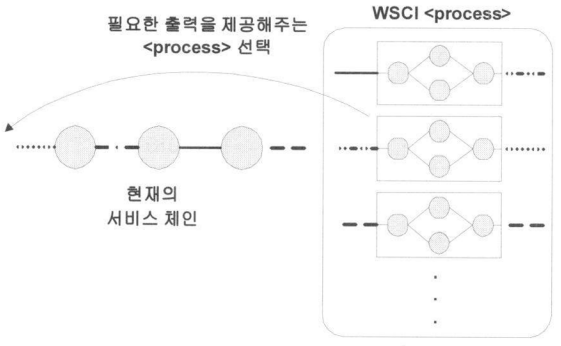
앞에서 설명했듯이 Choreography Analyzer에서는 WSCI의 <process>에 대한 정보를 생성한다. 각각의 <process>에 대한 정보는 그 자체가 하나의 서비스 체인이라고 볼 수 있으며, WSCI에 의해서 기술된 것이므로 이미 그 동작이 보장된 서비스 체인이라고 할 수 있다. 따라서 첫 번째 단계로 WSCI의 <process> 검색을 통해 필요로 하는 출력을 제공해주는 <process>(서비스 체인)를 찾는다. (그림 5)는 이러한 과정을 보여주는 그림이다.

3.3.3.2 WSCI의 <sequence> 검색

Choreography Analyzer에서는 WSCI의 <process>에 대한 정보 외에 모든 <sequence>에 대한 정보도 생성한다. 이 정보 역시 WSCI에 의해서 기술된 것이므로 여기서 나타내는 순차적인 수행 상에 연결된 서비스들은 그 연결 동작이 어느 정도 보장된 것이라고 볼 수 있다. 따라서 두 번째 단계에서는 WSCI의 <sequence>를 참고하여 이전에 연결될 서비스를 찾는다. 또한 하나의 <sequence>는 그 동작이 보장된 것이므로, 가능하면 같은 <sequence>에 있는 서비스들을 연속적으로 사용하면 컴포지션의 정확성을 높일 수 있다. (그림 6)는 이러한 과정을 보여주는 그림이다.



(그림 4) 기존 알고리즘에서 서비스 검색

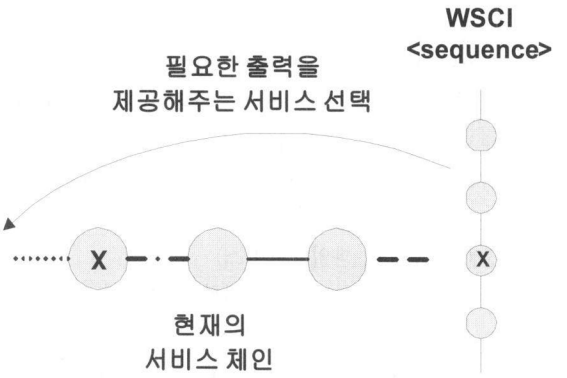


(그림 5) 제안 알고리즘에서 서비스 체인 검색

3.3.3.3 기존과 같은 방법으로 검색

마지막으로 세 번째 단계에서는 기존의 알고리즘과 같은 방법을 사용한다. 레지스트리 검색을 통해 필요로 하는 출력을 제공해주는 서비스를 찾는다.

다음의 (그림 7)은 본 논문에서 제안하는 자동 컴포지션 알고리즘이다.



(그림 6) 제안 알고리즘에서 서비스 검색

```
global Result
BackwardChaining(IN_req, OUT_req) {
    Need = OUT_req
    foreach p_i in CP {
        if Need ∩ OUT_p,i ≠ φ then
            Test(p_i, Need, IN_req)
    }
    if (w_first = first of Result) is single Web Service then {
        foreach s_i in CS (least recently used order) {
            if s_i have w_first then {
                foreach w_i in s_i (only before w_first) {
                    if Need ∩ OUT_s,i ≠ φ then
                        Test(w_i, Need, IN_req)
                }
            }
        }
    }
    foreach w_i in R {
        if Need ∩ OUT_w,i ≠ φ then
            Test(w_i, Need, IN_req)
    }
    return nothing
}
Test(f_i, Need, IN_req) {
    if Result not has f_i then {
        Result = f_i · Result
        Need = (Need − OUT_i) ∪ IN_i
        if IN_req ⊃ Need then
            output Result
        else
            BackwardChaining(IN_req, Need)
        remove f_i from Result
    }
}
```

(그림 7) 제안 알고리즘

3.3.4 BPEL프로세스 생성

Translator에서 하는 일은 컴포지션 결과를 실행 가능한 BPEL 프로세스로 변환하는 것이다. 컴포지션 결과는 3.1절에서 설명했듯이 <표 1>의 방법으로 표현된다.

4. 실험 및 평가

4.1 실험 데이터

실험을 위한 데이터로는 WS-Challenge[14]에서 제공하는 데이터를 사용하였다. 실험의 목적은 구체적인 BPEL 프로세스를 생성해내는지 여부와 자동 컴포지션의 정확성이 향상되었는지를 목적으로 하고 있다. 이를 위해 WS-Challenge 2006[14]에서 제공하는 데이터를 사용하였고 그 중 본 논문의 실험에서는 composition1-20-4를 사용하였다. composition1-20-4에서는 다음의 데이터가 제공되고 있다.

- 2156개의 WSDL 파일 (서비스)
- 11개의 사용자 요청과 이에 대한 컴포지션 결과

그러나 composition1-20-4에는 웹 서비스 코레오그라피 정보가 없기 때문에 실험을 위해 다음과 같이 composition1-20-4를 가공하여 사용하였다.

4.1.1 유효한 컴포지션 생성

각각의 사용자 요청 사항에 대해 역방향 추론 방법을 이용하여 composition1-20-4 데이터에 대한 자동 컴포지션을 수행하여 30개의 유효한 컴포지션을 구한다. 30개는 실험을 위해 임의로 정한 수이다.

4.1.2 코레오그라피 정보 생성

전 단계에서 구한 30개의 유효한 컴포지션 결과 중 70%인 21개가 커버되도록 코레오그라피 정보를 생성한다. 3.3.1절에서 설명한 바와 같이 코레오그라피 정보는 <process>에 대한 정보와 <sequence>에 대한 정보로 나누어지나 본 실험에서는 편의를 위해서 모든 <process>는 하나의 <sequence>로 이루어지도록 하였다.

따라서 실험을 위해 사용 또는 가공한 데이터 정보는 다음과 같다.

- composition1-20-4의 2156개의 WSDL 파일
- composition1-20-4에서 주어진 11개의 사용자 요청사항
- 각각의 "사용자 요청 사항"에 대해 30개의 유효한 컴포지션
- 유효한 컴포지션의 70%를 cover 할 수 있는 코레오그라피 정보 (수정된 WSDL 파일)

4.2 평가 방법

자동 컴포지션의 정확성을 평가하기 위한 방법으로 정확률(precision)과 재현율(recall)을 사용하였다[13]. 정확률과 재현율은 정보검색 분야에서 대표적으로 사용되고 있다. 정보검색에서는 대상이 되는 데이터 집합이 주어졌을 때, 사용자의 질의를 만족시키는 데이터 집합을 찾아내는 것이다. 유사하게 자동 웹 서비스 컴포지션에서는 단순히 문법적 정보에 의해서 조합 가능한 다양한 컴포지션 방법들 중에서 실제로 올바르게 수행되는 유효한 컴포지션을 방법을 찾아내는 것이 중요하다. 따라서 자동 웹 서비스 컴포지션의 정확성을 평가하기 위하여 정확률과 재현율을 사용할 수 있다.

4.3 실험 결과 및 평가

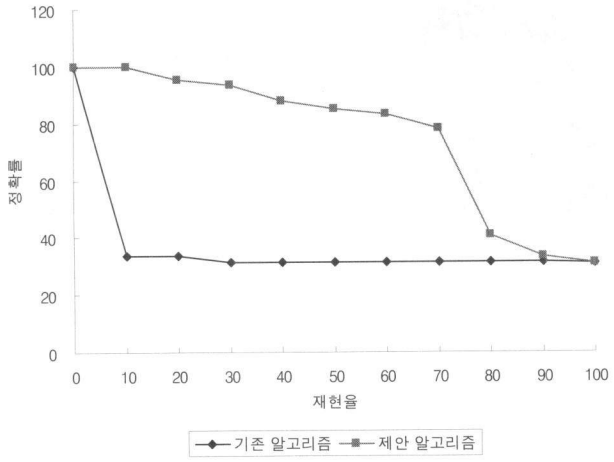
<표 2>는 정확성에 대한 실험 결과를 표로 나타낸 것이다. 4.1의 데이터에서 나오는 11개의 사용자 요청으로 자동 컴포지션을 수행하고, 그 결과의 정확률 평균을 구한 것이 평균 정확률이다. 정확률의 측정은 찾아진 컴포지션 결과들 중에서 미리 정해놓은 유효한 컴포지션 결과가 몇 개나 있는지 확인하여 유효한 결과의 비율로 계산하였다. 재현율은 유효 컴포지션이 얼마나 많이 재현되었는지를 나타낸다. 조정된 정확률은 정확률-재현율 그래프를 그리기 위하여 계산된 값이다. 비교를 위해서 [7], [8] 등에서 제시된 역방향 추론 방법 기반의 기존 알고리즘을 사용하여 함께 실험하였다.

(그림 8)은 <표 2>의 실험 결과를 정확률-재현율 그래프로 나타낸 것이다. 제안하는 알고리즘의 정확성이 큰 차이로 향상된 것을 볼 수 있다. 하지만 재현율이 70%를 넘어가면 정확성이 급격히 떨어지는 것을 볼 수 있다. 이는 유효한 컴포지션 결과 중 70%에 해당하는 부분에 대해서만 코레오그라피 정보를 만들었기 때문이다.

자동 웹 서비스 컴포지션에 웹 서비스 코레오그라피 정보를 이용함으로 컴포지션 결과의 정확성 및 구체성은 향상되었지만, 보다 다양한 정보의 관리 및 사용으로 인해서 컴포지션 시스템의 오버헤드는 증가하게 되었다. 제안하는 시스템의 또 다른 문제점은 WSCI로 기술된 코레오그라피 정보가 있는 범위에 한해서만 정확성 향상을 기대할 수 있다는 것이다. (그림 8)에서 재현율이 70%가 넘어가면 정확율이 급격히 떨어지는 것도 이러한 문제 때문이다. 하지만 현재의 웹 서비스들은 대부분 코레오그라피 정보를 갖고 있지

〈표 2〉 정확성 실험 결과

	기존 알고리즘		제안 알고리즘	
재현율	평균 정확률	조정된 정확률	평균 정확률	조정된 정확률
10	0.30	0.34	1.00	1.00
20	0.34	0.34	0.95	0.95
30	0.31	0.31	0.93	0.93
40	0.30	0.31	0.88	0.88
50	0.31	0.31	0.85	0.85
60	0.28	0.31	0.83	0.83
70	0.29	0.31	0.78	0.78
80	0.30	0.31	0.40	0.40
90	0.31	0.31	0.33	0.33
100	0.31	0.31	0.31	0.31



(그림 8) 실험 결과의 정확률-재현율 그래프

않기 때문에 제안하는 시스템으로 자동 웹 서비스 컴포지션 하기 위해서는 추가적인 코레오그라피의 기술이 필요하다.

5. 결론 및 향후 연구 과제

본 논문에서는 웹 서비스 코레오그라피를 이용한 자동 웹 서비스 컴포지션 시스템을 제안하였다. WSCI로 기술된 웹 서비스 코레오그라피 정보를 이용하여 자동 웹 서비스 컴포지션을 수행함으로 컴포지션 결과의 정확성을 향상시켰고, 보다 구체적이 컴포지션 결과를 만들어낼 수 있었다. 제안하는 시스템에서는 컴포지션의 결과로 조건 분기, 병렬수행 등의 다양한 수행 흐름이 표현되는 실행 가능한 BPEL 프로세스를 생성해낸다.

하지만 본 논문에서 제안한 자동 웹 서비스 컴포지션 시스템에는 다음과 같은 한계점도 있었다.

첫 번째로 코레오그라피 정보가 있는 범위에 한해서만 정확성 향상을 기대할 수 있다는 것이다.

두 번째, 한계점은 사용자 요청의 기술에 있어서는 단순히 문법적인 정보밖에 제공하지 못한다는 것이다. 컴포지션 되는 서비스들 사이의 연결은 코레오그라피 정보를 통해서 정확성을 높였지만, 사용자 요청과 서비스 사이의 연결은 이전의 문제가 그대로 남아있다.

이러한 한계점에도 불구하고 본 논문이 기여한 바는 웹 서비스 코레오그라피를 이용한 자동 웹 서비스 컴포지션이라는 새로운 접근방법을 제시하였고, 이를 위한 알고리즘 및 시스템을 제안한 것에 있다.

향후 연구 과제로는 사용자 요청을 기술하기 위한 새로운 방법 개발, 컴포지션 결과가 올바른 것인지 자동으로 검증하는 방법 개발 등이 있다. 이들은 다른 관련 연구에서도 이슈가 되고 있는 문제이기도 한다. 또한 본 논문에서 제안한 시스템에 시맨틱 웹 기술도 적용시키면 보다 좋은 결과를 가져올 수 있을 것이라 기대된다.

참 고 문 헌

[1] N. Milanovic, M. Malek, "Current Solutions for Web Service Composition," IEEE Internet Computing, Vol.8, No.6, pp. 51-59, IEEE Computer Society, 2004.

[2] A. Alves, A. Arkin, "Web Services Business Process Execution Language," Public Review Draft, OASIS, 2006.

[3] S. Thakkar, C. Knoblock, J. Ambite, "Dynamically Composing Web Services from On-line Sources," Proceedings of 2002 AAAI Workshop on Intelligent Service Integration, pp.1-7, AAAI Press, 2002.

[4] S. Oh, D. Lee, S. Kumara, "A Comparative Illustration of AI Planning-based Web Services Composition," ACM SIGecom Exchanges, Vol.5, No.5, pp.1-10, ACM Press, 2005.

[5] K. Pu, V. Hristidis, N. Koudas, "A Syntactic Rule Based Approach to Web Service Composition," Proceedings of the 22nd International Conference on Data Engineering, pp. 31-40, IEEE Computer Society, 2006.

[6] M. Jung, S. Cho, "A Novel Method based on Behavior Network for Web Service Composition," Proceedings of the International Conference on Next Generation Web Services Practices, pp.122-127, IEEE Computer Society, 2005.

[7] L. Aversano, G. Canfora, A. Ciampi, "An algorithm for Web service discovery through their composition," Proceedings of the IEEE International Conference on Web Services, pp. 332-339, 2004.

[8] J. Liu, J. Cui, N. Gu, "Composing Web services Dynamically and Semantically," Proceedings of the IEEE International Conference on E-Commerce Technology for Dynamic E-Business, pp.234-241, 2004.

[9] J. Liu, C. Fan, N. Gu, "Web Services Automatic Composition with Minimal Execution Price," Proceedings of the IEEE International Conference on Web Services, Vol.1, pp.302-309, 2005.

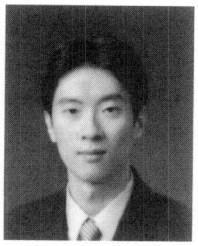
[10] S. Hashemian, F. Mavaddat, "A Graph-Based Approach to Web Services Composition," Proceedings of the 2005 Symposium on Applications and the Internet, pp.183-189, 2005.

[11] A. Arkin, S. Askary, S. Fordin, "Web Service Choreography Interface 1.0," W3C Note, W3C, 2002.

[12] N. Kavantzas, D. Burdett, G. Ritzinger, "Web Services Choreography Description Language Version 1.0," W3C Candidate Recommendation, W3C, 2005.

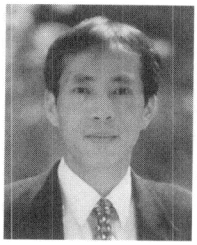
[13] R. Baeza-Yates, B. Ribeiro-Neto, "Morden Information Retrieval," Addison-Wesley, pp.75-81, 1999.

[14] The 2006 Web Services Challenge, http://insel.flp.cs.tu-berlin.de/wsc06/.



이 상 규

e-mail : sklee@ec.cse.cau.ac.kr
2005년 중앙대학교 컴퓨터공학과(공학사)
2007년 중앙대학교 대학원 컴퓨터공학과 (공학석사)
관심분야 : 웹 서비스, ebXML, mobile



한 상 용

e-mail : hansy@cau.ac.kr
1975년 서울대학교 공과대학(공학사)
1984년 Minnesota 공과대학(공학박사)
1984년~1995년 IBM 책임연구원
1995년~현 재 중앙대학교 컴퓨터공학과 교수
관심분야 : Web Engineering(Web Services, etc.), Information Retrieval, eCommerce Technologies