# Web API 발견 및 조합 기법

이용주
경북대학교 과학기술대학 컴퓨터정보학부
e-mail:yongju@knu.ac.kr

# Web API Discovery and Composition Techniques

Yong-Ju Lee
School of Computer Information, Kyungpook National University

요 약

최근 매쉬업에 대한 관심도가 매우 높아짐에 따라 수많은 Web API들이 생성되고 있다. 이런 Web API들을 매쉬업 속으로 결합할 때 여러 가지 이슈들이 존재한다. 특히, 수많은 API들이 매쉬업 개발자에 의해 수동으로 조합될 때 이는 더욱 심각해진다. 본 논문에서는 Web API 발견 및 조합을 위한 하나의 새로운 기법을 제안한다. 제안된 발견 기법은 질의를 만족시키지 못하는 API들을 재빨리 필터링 시키는 전략을 수립한다. API 조합 기법은 발견 기법을 확장/발전시켜 Web API 입출력 사이의 시맨틱 유사도를 기반으로 하고, 원하는 목표를 만족하는 출력을 산출할 수 있는 사이클 없는 방향성 그래프(DAG)를 생성한다. 또한, Web API 발견 및 조합을 효율적으로 생성하기 위해 본 논문에서는 Web API 발견 및 조합 시스템을 구현한다.

## 1. 서론

최근 웹 2.0이라는 말이 유행하면서 개방, 참여, 공유를 모토로 한 인터넷 환경이 급속도로 발전해왔다. 이러한 트렌드 속에서 다양한 웹 애플리케이션 상에서 호환성을 가지고 자원과 정보의 공유 및 상호작용이 필수적인 요소가 되었고, 이에 따라 SOA(Service Oriented Architecture) 및 웹 서비스 기술은 지속적으로 발전하고 있다[1]. 지금까지 웹 서비스는 주로 SOAP(Simple Object Access Protocol)을 이용해 구현되다가 최근에는 보다 단순하고 가벼운 REST(REpresentational State Transfer), JavaScript, XML-RPC, 그리고 Atom 기반의 Web API가 널리 이용되면서 서비스 제공자는 기존에 사용되어 왔던 SOAP 기반의 웹 서비스도 지원하면서 REST, JavaScript 등 다양한 프로토콜로 제공되는 Web API 방식도 제공할 필요성이 생겼다[2]. 또한, SOAP, REST, JavaScript, XML-RPC, 그리고 Atom에 대한 서비스 플랫폼을 따로 운영하는 것보다는 하나의 플랫폼 상에서 기존의 SOAP 서비스와 함께 다양한 Web API 서비스도 같이 지원하는 프레임워크를 요구하게 되었다[3].

본 논문에서는 이러한 이슈들을 해결하기 위한 기본 연구로써 그래픽 기반의 자동 Web API 발견 및 조합 기법을 제안한다. 간혹 사용자들은 API 저장소(repository)로부터 그들의 요구사항을 만족하는 API를 바로 발견(discovery)할 수도 있으나, 보통은 하나의 API 그 자체만으로는 다양하고 복잡한 사용자 요구사항들을 충분히 만족시켜 줄 수가 없기 때문에 API 조합(composition)을 통해 새롭고 유용한 솔루션을 만들 수 있다.

본 논문에서 제안되는 발견 알고리즘은 최종 결과물에 도움이 되지 않은 API들을 사전에 신속히 필터링하는 전략을 수립한다. 조합 알고리즘은 오퍼레이션 연결 그래프 구축 및 조합 후보군 탐색 과정으로 구성되어 있으며, 이의 최종 결과물은 사용자 요구사항을 만족시킬 수 있는 사이클이 없는 방향성 그래프(DAG: Directed Acycle Graph)의 생성으로 구현된다. 여기서 DAG들은 양방향 API 체이닝(chaining)에 의해 점차적으로 생성된다.

## 2. Web API 발견 및 조합 문제

### 2.1 API 발견 문제

하나의 질의와 Web API 집합이 주어졌을 때, 저장소로부터 질의 요구사항과 매치되는 오퍼레이션들을 탐색하는 과정을 Web API 발견 문제라 한다. 직관적으로 Web API에서 오퍼레이션들이 비슷한 입력과 출력을 가지고 있다면 이들 두 오퍼레이션은 유사하다고 할 수 있다. 예를 들어, 특정 지역의 위도와 경도를 제공하는 API를 검색하고 질의와 임의의 오퍼레이션에 대한 신택틱 정보는 <표 1>과 같다고 하면, 이 API는 우리의 질의를 만족한다. 왜냐하면, 질의 Q는 입력으로써 CountryCode, ProvinceName, 그리고 CityName을 요구하고 있고 출력으로써 Lat와 Long를 원한다. O은 입력으로써 CountryID

이 논문은 2010년도 정부(교육과학기술부)의 재원으로 한국연구재단의 기초연구사업 지원을 받아 수행된 것임(No. 2010-0008303).

와 NameOfCity를 취하지만, 여기서 Code와 ID는 같은 클러스터 개념으로 표현될 수 있고 CityName과 NameOfCity도 같은 객체(즉, City)의 속성으로 표현될 수 있으므로 같은 시맨틱으로 고려된다[4]. 따라서 Q의 입력을 받아 O를 수행할 수 있으며, 출력도 Q는 Lat와 Long를 원하기 때문에 O는 이를 만족한다.

<표 1> API 발견 문제

| 오퍼레이션 | 입력 | 출력 |
|---|---|---|
| Q | CountryCode<br>ProvinceName<br>CityName | Lat<br>Long |
| O | CountryID<br>NameOfCity | Lat<br>Long<br>Geography |

### 2.2 API 조합 문제

하나의 질의와 Web API 집합이 주어졌을 때, 저장소로부터 질의 요구사항과 매치되는 하나의 오퍼레이션을 발견할 수 없는 경우, 두 개 이상의 오퍼레이션들을 같이 결합시켜 원하는 목표를 만족시킬 수 있는 오퍼레이션 체인을 탐색해야 하는데, 이러한 것을 API 조합 문제라 한다. 즉, 어떤 오퍼레이션으로부터 생성되는 출력물이 다른 오퍼레이션의 입력으로 받아들여 질 수 있는 체인을 검색한다. 예를 들면, 특정 지역 날씨를 알 수 있는 오퍼레이션을 검색한다고 했을 때 그 결과는 <표 2>와 같다.

<표 2> API 조합 문제

| 오퍼레이션 | 입력 | 출력 |
|---|---|---|
| Q | AreaID<br>Province<br>CityName | Weather |
| $O_1$ | AreaCode,<br>NameOfCity | Lat<br>Long<br>Geography |
| $O_2$ | Lat<br>Long | Weather |

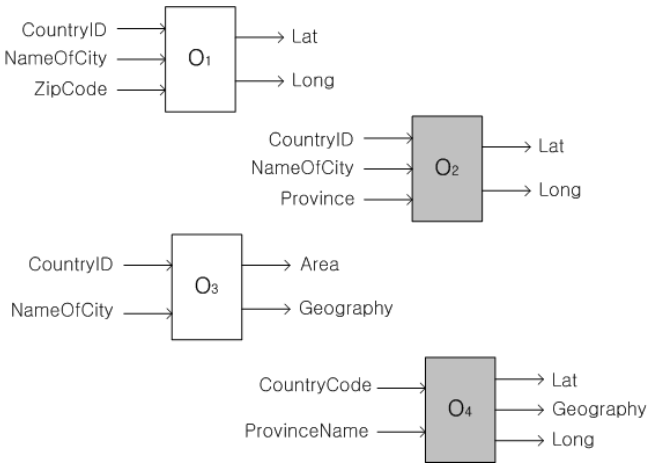
검색 엔진이 질의 요구사항을 만족할 수 있는 하나의 오퍼레이션을 발견할 수 없을 때, 이 엔진은 API 집합으로부터 다수의 오퍼레이션들을 조합할 수 있다. <표 2>에서 Q는 입력으로써 AreaID, Province, 그리고 CityName을 제공하고 출력으로써 Weather를 요구하고 있다. 오퍼레이션 $O_1$은 입력으로써 AreaCode와 NameOfCity를 요구하고 있는데, 여기서 비록 AreaID와 AreaCode는 다른 형태를 취하고 있지만 ID와 Code가 같은 클러스터 개념으로 표현될 수 있기 때문에 같은 시맨틱으로 참조될 수 있다. 또한 CityName과 NameOfCity도 같은 객체(즉, City)의 속성으로 표현될 수 있으므로 같은 시맨틱으로 고려된다. 따라서 Q의 입력을 받아 $O_1$을 수행할 수 있다. 한편, $O_1$은 출력으로써 Lat, Long, Geography를 리턴하는데, $O_2$는 이들 Lat, Long를 입력으로 받아 출력으로 Weather을 리턴할 수 있다. 따라서 후위의 $O_2$는 전위의 $O_1$에 의해 산출되는 출력물을 입력으로 사용하여 원하는 질의 결과를 산출한다.

이러한 형태의 Web API 조합 문제는 저장소로부터 하나의 연속적인 오퍼레이션 체인을 탐색하는 과정으로 정의할 수 있다. 구체적으로 설명하면, 조합의 첫 단계로 선택되어지는 오퍼레이션들은 질의의 입력만을 사용하여 수행할 수 있어야 하고, 조합의 마지막 단계로 선택되어 지는 오퍼레이션들은 그 출력들이 질의에서 요구되는 출력물을 모두 충족시켜야 한다. 그리고 조합을 구성하는 어떤 전위 오퍼레이션의 출력은 그 다음 후위 오퍼레이션의 입력으로 사용될 수 있어야 한다.

## 3. API 발견 및 조합 기법

### 3.1 API 발견 기법

API 발견 기법에 대한 전체적인 알고리즘은 다음과 같다. 질의 Q를 저장소에 있는 각각의 오퍼레이션 O와 비교한다. 먼저 Q의 매개변수를 사용하여 원하는 출력을 산출해 낼 수 있는 오퍼레이션들을 찾는다. 즉, 질의 출력 매개변수들을 오퍼레이션의 출력 매개변수들과 비교하여 출력 유사도를 계산하고, 매치가 실패하지 않는다면 반대로 오퍼레이션의 입력 매개변수들과 질의 입력 매개변수들을 비교한다. 그러나 질의 출력 매개변수들 중 하나라도 오퍼레이션 출력 매개변수들과 매치되지 않는다면 이 매치는 실패한다. 최종적으로 전체 유사도가 계산되고, 나중에 유사도 값에 대한 정렬을 수행하기 위해 매치된 오퍼레이션과 그의 유사도를 저장한다.




(그림 1) API 발견 기법에 대한 다양한 예

(그림 1)은 API 발견 기법에 대한 다양한 예들을 보이고 있다. 그림에서 $O_2$와 $O_4$는 API 발견 알고리즘을 만족한다. 그러나 $O_1$의 출력은 질의를 만족하지만 입력은 질의를 만족하지 못한다. 왜냐하면, 입력 중 ZipCode는 질의 입력 중에서 제공해 줄 수 없기 때문이다. $O_3$는 입력은 질의를 만족하지만 출력이 질의를 만족할 수 없다. 왜냐하면, 이의 출력이 질의 출력 Lat와 Long를 산출해 내지 못하기 때문이다.

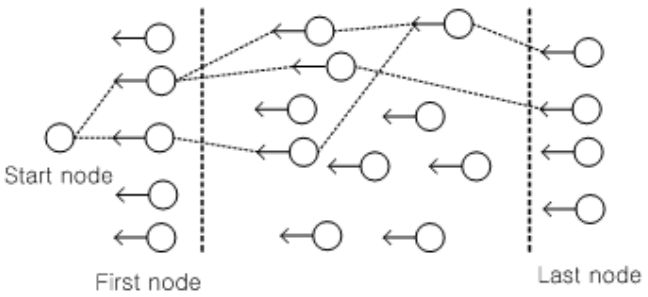
### 3.2 API 조합 기법

API 조합 기법을 구현하기 위해 본 연구에서는 API 발견 기법을 확장/발전시켰다. 본 기법은 주어진 노드로부터 목표 노드까지 최단 거리를 발견하기 위해 수정된 Dijkstra 알고리즘[5]을 사용한다. 제안된 기법은 다음과 같다.

먼저 질의의 출력 매개변수들을 모두 포함하고 있는 오퍼레이션들을 저장소로부터 탐색한다. 이들을 마지막 노드(last node)라 한다. 그리고 최소한 하나의 질의 입력 매개변수들을 가지고 있는 오퍼레이션들을 탐색한다. 이들을 첫 노드(first node)라 한다. 그 다음, 모든 마지막 노드들에 대해 그 노드에 연결된 노드들을 방문하여 트리들을 생성한다.

각각의 트리에 대해 하나의 시작 노드(start node)가 첨가된다. 시작 노드는 동적으로 생성되는 하나의 특수한 노드로써, 어떠한 입력 없이 질의의 입력을 이 노드의 출력물로 제공하는 노드이다. 따라서 하나의 가능성 있는 조합 후보를 찾는 것은 시작 노드로부터 시작하여 마지막 노드까지 연결된 하나의 DAG를 생성하는 것이다.

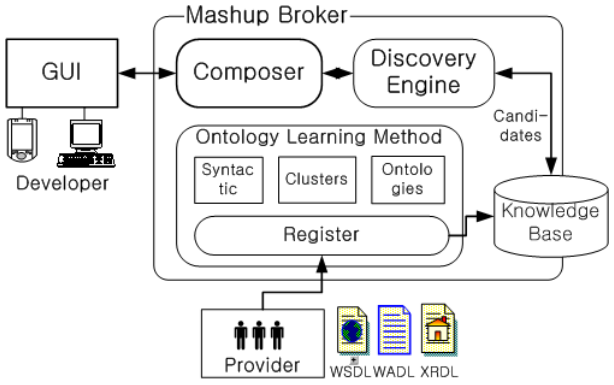
이러한 가능성 있는 조합 후보가 발견되었을 때, 마지막으로 조합에 참여하는 모든 간선들의 유사도 데이터를 수집한다. 유사도는 간선에 첨부되어 있는 모든 유사도 데이터의 평균값으로 계산되고, 조합 후보의 우선순위가 이 유사도에 의해 결정된다. 정렬된 조합 후보군 리스트 중 제일 위에 있는 결과가 사용자에게 가장 좋은 조합으로 추천된다. (그림 2)는 전체적인 API 조합 기법과 각 노드들을 설명하고 있다.




(그림 2) API 조합 그래프의 각 노드들

## 4. API 발견 및 조합 시스템 구현

API 발견 및 조합 시스템의 전체적인 구조는 (그림 3)과 같으며, 조합기(composer), 발견 엔진(discovery engine), GUI(Graphical User Interface), 그리고 온톨로지 학습 방법(ontology learning method)으로 이루어져 있다. 조합기는 원하는 목표에 대한 조합 계획을 수립하는 책임을 가진다. 이는 현재 조합 상태를 취합하고 매쉬업에 첨가될 수 있는 관련 API들을 동적으로 조합한다. 모든 조합 API들은 DAG 방식으로 처리되며, 정점(vertex)은 태스크를 표현하고 간선(edge)은 데이터 흐름을 나타낸다. 이 그래프는 G=(V, E), 즉 정점들의 집합 V와 간선들의 집합 E로 구성된다. 각각의 입력과 출력 간선들은 API의 입/출력을 나타낸다.




(그림 3) API 발견 및 조합 시스템 구조

발견 엔진은 API 발견이 요구되는 단계에서 사용자는 저장소로부터 적합한 후보 서비스들이 발견되도록 쿼리문을 작성하고, 발견 엔진은 이 쿼리를 기반으로 API 발견 과정을 수행한다. 발견된 API들은 조합을 이루는 개별 API들이 될 수도 있다. GUI는 마법사처럼 만들어져 있고, 여러 판넬에 의해 다음/이전 화면을 보여줄 수 있으며 사용자로부터 질의가 주어지면 API 조합 기법을 적용하여 이용 가능한 조합 리스트를 보여주고, 이들 중 가장 적합한 API 조합을 선택하도록 한다. 이러한 방식으로 마지막 결과가 만족될 때 까지 사용자들의 목표를 반복적으로 정제한다. 온톨로지 학습 방법[6]은 Web API를 개발할 때 자동으로 생성되는 WSDL/WADL/XRDL 문서만 가지고 항목간 숨어 있는 시맨틱 정보를 찾아내어 올톨로지를 자동 구축한다.

본 연구에서는 자동 API 조합 시스템을 구현하기 위해 그래프 기반 목표 지향 접근 방법을 적용하였다. 워크플로우 시작 단계부터 양방향 체이닝 알고리즘에 의해 자동적으로 DAG를 생성한다. 따라서 본 기법을 활용하면 어떤 특별한 통합 노력이나 프로그래밍 기술 없이 데이터 매쉬업을 자동적으로 만들 수 있는 이점을 제공한다.

## 5. 결론

본 논문에서 제안하는 Web API 발견 및 조합 기법은 매쉬업에 대한 자세한 처리 과정은 알 필요 없이 주어진 질의로부터 원하는 목표를 이룰 수 있다. 즉, 매쉬업 개발자는 질의의 형태로 원하는 목표를 간단히 기술한 후 시스템에 처리를 요구하고, 만일 원하는 결과가 하나의 오퍼레이션 출력에 직접 매치될 수 있다면 이는 API 발견 문제로 단순화될 수 있고, 그렇지 않으면 원하는 결과물을 산출할 수 있는 연속적인 오퍼레이션 체인을 탐색하여 조합 문제를 해결한다. 이러한 오퍼레이션 체인은 주어진 질의로부터 생성될 수 있는 하나의 DAG 검색 문제로 볼 수 있다.

본 논문에서 제안하는 API 조합 기법은 그래프 기반 접근방식에 기반을 두고 있으며, 조합 가능한 후보군들이 API 오퍼레이션 체인에 의해 점차적으로 생성된다. 또한, 본 기법에서는 질의를 만족시키지 못하는 API들을 재빨리 필터링 시키는 전략을 수립하여 하나의 최적 솔루션을 제공하고 있다.

## 참고문헌


[1] D. K. Barry, *Web Services, Service-Oriented Architectures, and Cloud Computing*, Elsevier, 2012. 12

[2] 최윤정, 차승준, 이규철, “다양한 프로토콜을 지원하는 시맨틱웹 기반 Open API 통합 프레임워크 개발,” 한국정보과학회 2010년 가을 학술발표논문집, Vol. 37, No. 2(C), 2010

[3] 김진한, 이병정, “웹 서비스와 Open API를 사용한 SOA 기반 동적 서비스 합성 프레임워크,” 정보과학회논문지: 소프트웨어 및 응용, 제36권, 제3호, 2009년 3월, pp. 187-199

[4] 이용주, “스마트 매쉬업을 위한 시맨틱 기반 Open API 온톨로지 구축 기법,” 디지털산업정보학회논문지, 제7권 제3호, 2011년 9월, pp. 11-23

[5] T. Cormen, C. Leiserson, R. Rivest, and C. Stein, *Introduction to Algorithms* (Second Edition), MIT Press, 2001.

[6] Y. J. Lee and J. H. Kim, “Semantically Enabled Data Mashups using Ontology Learning Method for Web APIs,” Proceedings of the 2012 Computing, Communications and Applications Conference, 2012