반응형 계획기를 이용한 자동화된 시맨틱 웹서비스의 조합

진 훈° 김인철
경기대학교 전자계산학과
{bioagent°, kic }@kyonggi.ac.kr

Automated Semantic Web Service Composition using a Reactive Planner

Hoon Jin°, InCheol Kim
Dept. of Computer Science, Kyonggi University

요 약

시맨틱 웹서비스 조합을 위해 인공지능 분야의 계획수립 기법을 적용하려는 연구들이 최근 늘고 있다. 하지만 전통적인 계획수립 기법이 갖는 계획작성 단계와 실행 단계가 분리되어 있기 때문에 여러가지 한계성을 나타낸다. 본 논문에서는 이의 대안으로서 동적 재연결을 지원하는 반응형 계획기 기법을 사용함으로써 제기된 문제점들을 해결하고자 하였으며, 온톨로지 처리 결과를 계획수립 과정에 반영함으로써 의미 기반의 웹서비스 조합이 이루어지도록 하였다. 그리고 이를 지원하는 시스템의 개발과 응용을 통해 반응형 계획기술이 시맨틱 웹서비스의 조합과정에서 견고하고, 유연하게 동작함을 알 수 있었다.

1. 서론

시맨틱 웹서비스란 웹서비스 기술에 의미적인 주석을 추가하고 주석 정보에 대해 온톨로지 처리를 함으로써 의미 기반의 검색, 발견, 조합, 실행, 감시 등의 기능을 제공할 수 있는 서비스를 말한다. 특히 서비스 조합과 관련되는 연구들이 최근 들어 인공지능 연구자들을 중심으로 증가하고 있으며, 이들은 주로 시맨틱 웹서비스의 명세상의 특징을 이용해 오랫동안 연구되어 온 계획수립 기법을 적용하려고 시도하고 있다. 그러나 전통적 계획수립 기법은 계획작성 단계와 실행 단계가 분리되어 있어서 시맨틱 웹서비스의 조합 과정에 바로 적용하는 데에는 여러가지 한계에 부딪히게 된다. 이러한 문제점들로는, 실행과정을 고려하지 않고 계획을 작성함으로 인해 실제 실행 시에 발생할 수 있는 여러가지 가변성과 불확실성, 에러 상황 등에 대해 대처할 수 없다는 것과, 역으로 실행과정을 고려하고자 하여도 실행 시에 발생 가능한 모든 상황들을 미리 고려하여 계획을 수립할 수 없다는 것이다. 이에 우리는 시맨틱 웹서비스 조합문제를 인공지능 분야의 계획수립 기법 중, 전통적인 계획수립 기법이 갖는 한계성에 대한 대안으로서 반응형 계획기 기술을 사용하기로 하였다. 반응형 계획수립 기법 역시 실제의 서비스 조합 과정에 적용하기 위해서는 적절한 가정과 전략이 필요하며, 본 논문에서는 이를 위한 방법론을 제시하였다. 그리고 프로토타입 시스템을 개발하여 개인들을 위한 가상의 보건 의료 시스템에 적용함으로써 반응형 계획기 기술이 시맨틱 웹서비스 조합에 적합하게 사용될 수 있음을 확인할 수 있었다.

2. 관련 연구

2.1 웹서비스 조합

웹서비스는 소프트웨어 요소들의 재사용성과 분산된 응용 소프트웨어들의 통합을 지원하며 SOAP, WSDL, UDDI 기술들로 구성된다. 웹서비스 조합에 관한 연구들을 살펴보면, 기업들을 중심으로 한 연구들과 시맨틱웹 연구자들을 중심으로 이루어진 연구들로 나뉠 수 있다. 전자의 연구결과로는 XLANG, WSFL, BPEL4WS, WSCI 등이 존재하고, 후자의 경우 의미적 주석과 온톨로지를 이용하는 DAML-S(또는 OWL-S) 명세를 기반으로 조합문제를 해결하기 위한 인공지능 분야의 연구들과 그 외의 연구 등이 있다. BPEL4WS를 중심으로 하는 기업형 연구들이 WSDL을 이용하여 주로 비즈니스 프로세스 관점에서 전문가에 의해 수동으로 디자인된 서비스 흐름에 따라, 서비스를 모델화하고 역할에 따라 그들 간의 상호작용이나 인터페이스를 정의하는 내용을 다루는데 비해, 후자의 연구는 DAML-S(OWL-S) 명세를 각각의 기법에 맞게 정보를 추출하거나 변환과정을 수행한 후, 이를 대상으로 의미적인 기반의 서비스 조합과정을 달성하는 방법들을 취한다. PDDL기반으로 주어진 태스크를 계층적으로 분해하는 방식을 취하는 HTN, 그리고 PDDL을 이용한 하이브리드 계획기에 관한 연구인 OWLS-Xplan 등이 이에 속한다. 이와 같은 연구들은 조합방식에 있어서 사람의 개입여부에 따라 수동과 자동으로 나뉘고, 조합된 서비스를 실행하는 측면에서 동적인 재연결을 지원하는지 여부에 따라 정적과 동적 방식으로 나뉠 수 있다.

표 1 시맨틱 웹서비스 조합 연구 분류

	수동	자동
정적	WSDL	HTN, OWLS-Xplan
동적	BPEL4WS, WSCI	?

2.2 반응형 계획 기술

반응형 계획 구조는 에이전트 내부에 실세계에 대한 아무런 기호모델을 갖지 않고 복잡한 기호 추론과정도 사용하지 않는 구조이다. 대신 반응형 계획 구조는 기호대신 실세계 정보들에 대해 센서를 통해 감지한 신호 그 자체를 처리함으로써 곧바로 행동을 결정한다. 따라서 환경 변화라는 자극에 대해 빠르게 반응할 수 있고 들어오는 입력신호를 인자로 취하는 간단한 함수나 회로로 구현 가능하므로 비교적 구현이 용이하다는 장점을 갖는다. 반응형 계획 구조를 제공하는 구현물로는 PRS(JAM), JACK, NUIN 등이 있다. 특히 JAM은 BDI구조로서 작은 크기의 시스템이라는 장점을 지니며, 구성 요소로서 월드 모델, 계획라이브러리, 인터프리터, 의도구조, 감시기를 갖는다 [3]. 하지만 상태정보를 기반으로 하여 추론과정을 전개하는

한계성과, 미리 부분적 계획 정보인 단위 계획들로 이루어진 추상화된 계획을 가져야만 한다는 단점을 갖는다.

3. 반응형 계획기를 이용한 웹서비스 조합

3.1 가정과 전략

시맨틱 웹서비스는 인터넷 환경에서 동작하기 때문에 독립성, 분산성, 가변성과 불확실성이라는 성질을 가지면서도, 실시간적으로 응답을 요구하는 특성을 지닌다. 또한 온톨로지를 이용하여 의미 기반의 서비스 수행을 지원할 수 있다. 그러므로 계획수립 기법을 사용하여 시맨틱 웹서비스들을 조합하기 위해서는 수립 과정에서 우선 서비스들이 인터넷 환경에서 동작함을 고려하고, 다음으로 서비스들 간의 연결이 온톨로지에 기반한 의미적 추론을 따르도록 해야 한다. 이와 같은 요구 사항들을 따르며 시맨틱 웹서비스 조합이라는 문제를 해결 하기 위해 우리는 서비스 명세 언어로서 OWL-S를 사용하고자 하였다. 그리고 첫째 요구사항에 대한 대책으로서 반응형 계획 기술을 사용하기로 하였다. 이는 반응형 계획 기술이 빠르게 동작하고, 외부 환경 변화를 인식하여 이에 능동적으로 대처하는 재계획 과정을 지원하며, 또한 비교적 구현작업이 쉽기 때문이다. JAM은 반응형 계획 기법을 지원하는 시스템으로서 목표 지향형 계획 과정을 수행하고, 문제 해결을 위해 계층적으로 하부 목표를 수행하는 특징을 갖는다[3]. 그러므로 시맨틱 웹서비스의 조합 문제에 대해 계획수립 기법을 적용하는 것은 OWL-S 서비스들에 대해 JAM을 이용하여 의미적 추론 결과에 따라 계획과정을 수행하는 것으로 간주될 수 있다. 조합과정을 달성하기 위해 우선적으로 OWL-S로부터의 모든 정보들을 이용하여 JAM 단위 계획 명세를 생성해야 한다. 이는 명세 변환 과정을 통해 이루어질 수 있는데 유사한 접근방법을 갖는 연구들로는 [4,2]가 있다. 그러나 OWL-S 명세와 JAM 단위 계획 명세는 적용 도메인의 차이로 인해 근본적으로 다르다. 그러므로 원활한 웹서비스 조합과정을 달성하기 위해 몇 가지 가정들이 필요하며 이를 [표 2]에서 나타내었다. 두번째 경우를 위한 대책으로서 [2]에서 온톨로지 정보들을 계획수립을 위한 명세언어인 PDDL의 문제기술 명세와 영역지식 명세에 담아내려는 시도가 있었지만 아직까지는 완전치 않은 수준이다. 그러므로 우리는 OWL-S로부터 추출된 용어의 타입정보들로부터 시맨틱웹 차원의 온톨로지 처리를 통한 결과를 계획과정 중에 반영토록 함으로써 의미적인 서비스 조합이 이루어지도록 하였다.

표 2 웹서비스 조합을 위한 가정

(1)	OWL-S 프로세스 모델인 K={K_1,K_2,...,K_n}이 주어질 때,
	(ㄱ) K집합 내에 존재하는 모든 단일 프로세스들은 출력 결과 또는 출력 후 조건 중 하나만 가진다.
	(ㄴ) K집합 내의 모든 복합 프로세스들은 단일 프로세스로 간주되어 조합과정에 이용될 수 있다.
(2)	사용자로부터의 요구 사항(질의) 입력은 제공되는 목표 리스트로부터 선택과정을 통해서 이루어진다.
(3)	K집합 내의 임의의 K가 2개 이상의 출력을 내는 경우 각각의 경우에 대해 목표로 설정하는 단위 계획이 만들어진다.
(4)	K집합 내의 임의의 K_n과 K_{n+1}이 순서대로 연결하고자 할 때 K_n의 출력인수 개수와 K_{n+1}의 입력인수 개수는 같다.
(5)	온톨로지를 이용한 의미 분석은 스키마 정보에 한한다.

3.2 시스템 구성

상기 기술한 내용에 따라 우리는 시맨틱 웹서비스 조합기로서 SWEEP 시스템을 설계하였다. SWEEP은 핵심 구성요소로서 반응형 엔진 처리기와 웹서비스 명세들로부터 자동으로 단위 계획 명세를 생성하는 OWL-S2JAM 계획 변환기를 가진다. 또한 변환작업을 위해 OWL-S로부터 추상화된 정보를 추출하여 저장하는 서비스 명세 처리기와 온톨로지 추론과정을 통해 서비스 매칭 결과를 알려주는 서비스 연결 처리기가 있으며 이 외에도 실행 관리기, 웹서비스 호출기 등이 있다.

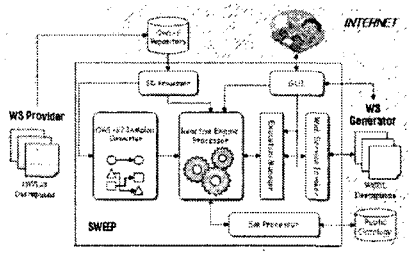

그림 1 SWEEP 구조

3.3 웹서비스의 예

명세 변환 과정을 위해 우리는 가상의 보건 의료 서비스로서 질병 검색, 진료과 검색, 의사 검색, 병원 검색, 진료 예약 서비스들을 개발하였다. 그리고 반응형계획 기술의 적용 가능성을 확인하기 위해 동일한 목표를 갖지만 에러를 일으키거나 의미적인 타입 정보를 달리하는 서비스를 추가로 개발하였다.

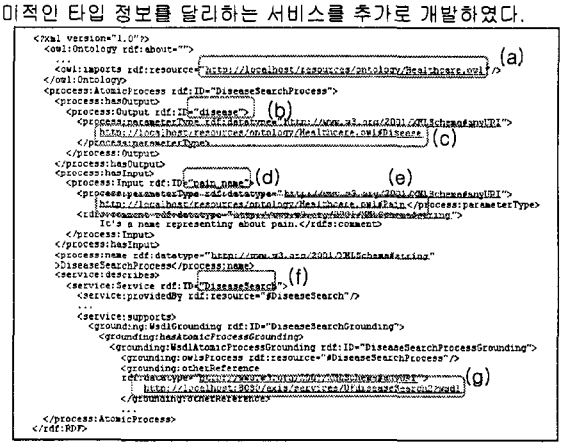

그림 2 질병 검색을 위한 OWL-S

[그림 2]에서 (a)~(g)는 각각 온톨로지 URI, 입력 변수명, 입력 변수타입, 출력 변수명, 출력 변수타입, 서비스명, WSDL URI를 나타낸다.

3.4 서비스 검색과 명세 변환

[표 2]에서 정의한 가정에 따라 입력된 OWL-S 명세들은 GUI로부터 사용자의 선택과 변환 행위, 그리고 추상계획 생성 행위에 의해서, 각각 JAM 단위 계획 생성 및 실제 계획수립을 위한 준비단계에 도달하게 된다. 계획수립 과정은 반응형 계획 기술의 특성 상 주어진 목표를 달성하기 위해 하부 목표 수행을 위한 과정을 포함한다. 이때 하부 목표를 달성하는 수단은 OWL-S로부터 변환된 계획들이며, 복수 개가 존재할 수 있다. 그러므로 동일 목표를 가진 복수 개의 계획들로부터 현재의 목표 달성을 위해 취해져야 할 하부 계획을 선택하는 작업이 필요하다. 선택작업은 즉시로 행해져야 하며, 추상 계획을 구성하는 모든 단위 계획들로부터 검색되어야 한다. 또한 이 과정에는 OWL-S명세로부터 타입 정보([그림 2]의 (c), (e))들에 대한 온톨로지 추론 결과가 반영되어야 한다. 하지만 실시간으로 OWL-S와 JAM단위 계획 명세들로부터 정보를 검색하고, 검색된 정보에 대해 온톨로지 처리를 반영하기 위해서는 오버헤드

가 따르게 되고, 그러므로 별도의 객체 정보를 생성하여 이용하는 것이 필요하다. 이에 본 연구에서는 ASDO(Abstract Service Description Object)를 생성하였고 서비스명세 처리기(SD processor)를 통해 생성되고 이용되도록 하였다. ASDO는 MIndSwap의 owl-s 1.1 api를 이용하여 구성된다.

표 3 추상화된 서비스 명세 객체(ASDO)

```
//목표명,서비스명,입력인수개수,입력변수명(parameterType명),출
력인수개수,출력변수명(parameterType명),wsdl주소
disease_known,Disease
Search,1,pain_name(http://localhost/resources/ontology/Healt
hcare.owl#Pain),1,disease(http://localhost/resources/ontology
/Healthcare.owl#Disease),http://localhost:8080/axis/services/
UFdiseaseSearch?wsdl
...
```

ASDO를 이용하여 실제로 OWL-S명세로부터 JAM단위 계획으로 변환하는 원리는 [표 4]에 기술하였다. 특히 목표명의 경우 OWL-S명세의 출력으로부터 계획에서의 목표 상태를 나타내기 위해 " 출력_known" 형태로 표기하였다. 이렇게 변환된 계획 명세는 [그림 3]에서 나타내었다. [그림 3]의 (b)' ,(f)' ,(d)' ,(g)' 는 각각 [그림 2]의 (b),(f),(d),(g)로부터 유도되어 작성된다.

표 4 명세 변환 원리

Translate Output(Q) of Atomic-Process to Goal
- *process:hasOutput rdf:ID="disease"*
- GOAL: disease_known "True"

Translate ServiceName(Q) of Atomic-Process
- *service:Service rdf:ID="Disease Search"*
- NAME: "Disease Search".

Translate Input(Q) of Atomic-Process
- *process:hasInput rdf:ID="pain_name"*
- BODY: ASSIGN $param (primitives.RequestUserInput.execute $param);
  ASSIGN $pain $param;
- BODY: ACHIEVE pain_name_known "True";
  RETRIEVE pain_name $pain;

Translate WSDL(Q) of Atomic-Process
- *<grounding:wsdlDocument>http://localhost:8080/axis/services/UFdiseaseSearch?wsdl*
  </grounding:wsdlDocument>
- BODY: ASSIGN $service_WSDL http://localhost:8080/axis/services/UFdiseaseSearch?wsdl;

Translate Output(Q) of Atomic-Process
- *process:hasOutput rdf:ID="disease"*
- BODY: EXECUTE primitives.ServiceCall $service_name $service_WSDL $pain $disease;
  ASSERT disease $disease.

3.5 실행 및 재계획

지금까지 설명한 내용을 바탕으로 실행과정을 설명하면 다음과 같다. 시스템은 사용자로부터 선택된 진료예약이라는 목표를 설정하고, 목표 달성을 위해 요구되는 IOPE를 고려하여 사용자로부터의 입력정보를 기대하며, 없을 시에 의료전문가가 근무하는 병원명을 출력으로 하는 서비스를 검색하게 된다. 병원 검색 서비스는 다시 사용자로부터 검색을 위한 입력 인수인 해당 질병 전문가명의 입력을 기대하며 없을 시에는 해당 질병 전문가명을 출력으로 하는 서비스를 찾는다. 이러한 방식으로 진료과 검색, 질병 검색 과정이 이루어지며, 최종적으로 환자의 증상을 입력으로 하는 서비스를 검색해냄으로써, 증상 정보만을 입력하여 절차적으로 진료예약 서비스를 위한 각 서비스들을 수행할 수 있게 된다. 각 서비스들은 수행 초기에 사용자로부터의 입력정보를 요청함으로써 계속적으로 하부 서비스로 분기하는 과정을 단축할 수 있다. 또한 반응형 계획 기법의 특성에 따라, 에러가 발생 시에는 동일 목표를 달성할 수 있는 대체 서비스를 구동시킬 뿐만 아니라, 필요할 경우 재계획과정을 통해 안정적으로 사용자의 요구를 충족시킬 수 있다.

4. 구현 및 응용

[그림 3]은 질병 검색 서비스를 위한 OWL-S 명세에 대해 구현된 JAM 계획 명세이고, SWEEP 시스템을 가상의 보건의료 서비스에 적용하여 실험한 결과를 [그림 4]에서 나타내었다.

```
plan: (
NAME: "Disease Search"                                    (b)'
GOAL: ACHIEVE disease_known "True";
PRECONDITION: RETRIEVE disease_known $known; (== $known "False");
BODY:
    ASSIGN $service_name "Disease Search";
    ASSIGN $currServ_paramNum 1;                          (f)'
    RETRIEVE simul_mode $smode;
    OR
    (   TEST(== $smode "OFF");

        ASSIGN $cmt_param "Pain: ";
    ASSIGN $param (com.irs.jam.primitives.RequestUserInput.execute $cmt_param $param);
        WHEN: TEST (!= $param "") ( ASSIGN $pain_name $param; );
        WHEN: TEST (== $param "") ( ASSERT pain_name "Unknown"; );   (d)'
            DO( OR( ACHIEVE pain_name_known "True";
                    RETRIEVE pain_name_known $known; )
                  ( EXECUTE println "Subgoaling Error!"; );
                  ) WHILE: TEST(!= $known "True" );
              RETRIEVE pain_name $pain_name; );
    ASSIGN $service_WSDL "http://localhost:8080/axis/services/DiseaseSearch?wsdl";
      EXECUTE com.irs.jam.primitives.ServiceCall $service_name $service_WSDL
      $pain_name $disease;
      WHEN: TEST (!= $disease "") ( ASSERT disease $disease; ); )
    (   TEST(== $smode "ON");                                    (g)'
        DO( OR( ACHIEVE pain_known "True";
                RETRIEVE pain_known $known; )
              ( EXECUTE println "Subgoaling Error!"; );
            ) WHILE: TEST(!= $known "True");
        SUCCEED; );
EFFECTS:
    EXECUTE com.irs.jam.primitives.GetCurrentPlanName.execute $service_name $planList;
    UPDATE (disease_known)(disease_known "True");
)
```

그림 3 질병 검색을 위한 JAM 단위 계획

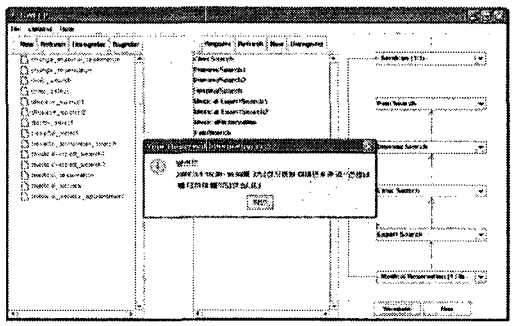

그림 4 SWEEP 실행 화면

5. 결론

우리는 본 논문에서 시맨틱 웹서비스 조합이라는 문제를 대상으로 인공지능 계획수립 기법을 적용하여 해결하고자 하였다. 전통적인 인공지능 계획수립 기법의 한계로 인해 대안으로서 반응형 계획기 기법을 적용하였으며 이의 적용을 위한 방법론을 기술하였다. 응용을 위해 개발된 SWEEP을 통해 가상의 보건의료서비스에 적용한 결과 서비스 오류 및 에러 상황에 대해 서비스 대체 기능과 재계획과정을 통해 사용자 입장에서 볼 때 안정적인 서비스를 제공받을 수 있음을 확인하였다.

참고 문헌

[1] Joachim Peer, "Web Service Composition as AI Planning - a Survey", Technical Report, Univ. of St.Gallen, 2005.

[2] Klusch, M., Gerber, A., Schmidt, M., "Semantic Web Service Composition Planning with OWLS-XPlan", *Proc. of the 1st Intl. AAAI Fall Symposium on Agents and the Semantic Web*, Arlington VA, USA, 2005.

[3] Marcus J. Huber, "JAM: A BDI-theoretic Mobile Agent Architecture", *Proc. of the 3rd Intl. Conference on Agents'99*, pp236-243, Seattle, WA, May 1999.

[4] Sirin E, Parsia B, Wu D, Hendler J, Nau D, "HTN Planning for Web Service Composition Using SHOP2", Jrnl of Web Semantics, Vol.1, No.4, pp.377-396, 2004.