단순 RDF 메시지의 온톨로지 상호 운용성을 위한 변환 규칙들의 연쇄 조합

(Cascade Composition of Translation Rules for the Ontology Interoperability of Simple RDF Message)

김 재 훈 † 박 석 ††
(Jaehoon Kim) (Seog Park)

요 약 최근 모바일과 유비쿼터스 컴퓨팅에서 보다 지능적인 다양한 서비스를 제공하고자 하는 비즈니스 전략과 함께 온톨로지 기술이 큰 관심이 되고 있다. 온톨로지를 이용하는 응용 도메인에서의 본질적 문제점은 모든 영역 구성원, 에이전트, 응용 프로그램이 온톨로지에서 정의된 동일 개념을 공유해야 하는 것이다. 하지만, 다양한 제조업자에 의해서 만들어지는 다양한 모바일 디바이스, 센싱 디바이스, 네트워크 구성요소, 다양한 통신 사업자, 다양한 서비스 제공업자 들이 모여 이루어지는 모바일과 유비쿼터스 컴퓨팅 환경에서는 서로 상이한 온톨로지가 공존할 가능성이 높다. 이러한 의미적 상호 운용성의 문제를 해결하고자 했던 다수의 연구가 있다. 이를 크게 분류하면, 맵핑, 합병, 변환에 의한 방법들이다.

본 연구에서는 이러한 방법들 중 OntoMorph와 같이 상이한 온톨로지 데이타들 간에 변환 규칙을 직접 작성하여 사용하는 방법에 초점을 맞춘다. 하지만 이러한 변환 규칙을 수작업으로 직접 작성하는 방법은, 그 자체도 어려울뿐더러 N개의 온톨로지가 존재할 경우 최악의 경우 $O(N^2)$의 변환 규칙 작성 복잡도를 갖는다. 따라서 본 논문에서는 이러한 복잡도를 개선하기 위한, 웹의 개방성에 근거한 연쇄 조합 변환 규칙 생성의 개념을 소개한다. 연구 성과는 변환 규칙의 변환의 신속성, 변환의 적합성, 변환 규칙 작성의 용이성 등의 중요한 평가 요소를 도출할 수 있었으며, 몇 가지 실험 및 기존 연구와의 비교 분석을 통하여 제안된 방법이 신속성과 정확성을 보장하면서 보다 높은 용이성을 가짐을 확인할 수 있었다.

키워드 : 온톨로지 상호 운용성, 온톨로지 변환, 연쇄 조합, XQuery 중첩

Abstract Recently ontology has been an attractive technology along with the business strategy of providing a plenty of more intelligent services. The essential problem in application domains using ontology is that all members, agents, and application programs in the domains must share the same ontology concepts. However, a variety of mobile devices, sensing devices, and network components manufactured by various companies, a variety of common carriers, and a variety of contents providers make multiple heterogeneous ontologies more likely to coexist. We can see many past researches fallen into resolving this semantic interoperability. Such methods can be broadly classified into by-mapping, by-merging, and by-translation.

In this research, we focus on by-translation among them which uses a translation rule directly made between two heterogeneous ontology data like OntoMorph. However, the manual composition of the direct translation rule is not convenient by itself and if there are N ontologies, the direct method has the rule composition complexity of $O(N^2)$ in the worst case. Therefore, in this paper we introduce the cascade composition of translation rules based on web openness in order to improve the complexity. The research result made us recognize some important factors in an ontology translation

· 본 연구는 한국과학재단 특정기초연구(R01-2006-000-10609-0) 지원으로 수행되었음

† 정 회 원 : 서강대학교 컴퓨터학과 연구교수
chris3@sogang.ac.kr
†† 종신회원 : 서강대학교 컴퓨터학과 교수
spark@dblab.sogang.ac.kr

논문접수 : 2006년 12월 18일
심사완료 : 2007년 9월 10일

정보과학회논문지: 데이타베이스 제34권 제6호(2007.12)

system, that is speediness of translation, soundness of translation, and conveniency of translation rule composition, and some experiments and comparing analysis with existing methods showed that our cascade method has more conveniency with insuring the speediness and the correctness.

Key words : Ontology Interoperability, Ontology translation, Cascade Composition, Nested XQuery

1. 서 론

RDF(Resource Description Framework)[1]와 OWL (Web Ontology Language)[2]은 웹 온톨로지 표준 언어로 시맨틱 웹을 위한 두 가지 핵심 기반 기술이다. 또한 웹에서 뿐만이 아니라, 최근 모바일과 유비쿼터스 컴퓨팅에서의 보다 지능적인 풍부한 서비스를 제공해야 하는 요구 사항과 함께 그 관심이 더욱 커지고 있다. 예로, OMA(Open Mobile Alliance)의 UAProf(User Agent Profile)[3]는 모바일 디바이스의 지원 사양(capabilities), 운영 환경, 네트워크 환경, 그리고 콘텐츠 사용에 관한 사용자의 관심을 기술할 수 있는 CPI (Capabilities and Preference Information)를 정의하며, 이러한 CPI 데이타에 따라 콘텐츠 제공자 혹은 네트워크 구성 요소들은 디바이스에 적합한 양질의 서비스와 통신을 제공할 수 있다. UAProf는 RDF 언어로 기술된다. 즉, 사용자가 특정 휴대 단말기로 음악 서비스를 제공하는 어떤 콘텐츠 서버에 접속할 경우, 그 휴대 단말기의 음원 지원이 64 화음이라는 CPI 정보를 통하여, 콘텐츠 서버는 64 화음의 음악 파일 리스트만이 들어있는 웹 페이지를 동적으로 만들어 전송할 수 있다. 또한 최근 유비쿼터스 컴퓨팅에서의 핵심 기술 중에 하나인 상황 인식(context-aware) 기술에 대한 많은 연구와 구현이 이루어지고 있는데, 상황 정보를 표현하는 온톨로지 언어로 RDF의 확장인 OWL을 사용하는 프로젝트가 대다수 있다: SOUPA[4], CoBra[5].

하지만 RDF와 OWL 응용의 본질적 문제점 혹은 전제는 특정 영역에서의 온톨로지 어휘(이러한 어휘는 온톨로지의 개념인 클래스와 속성을 표현)들을 영역 구성원, 에이전트, 응용 프로그램들이 공유해야 된다는 것이다. 하지만 다양한 제조업자에 의해서 만들어지는 다양한 모바일 디바이스, 센싱 디바이스, 네트워크 구성요소, 다양한 통신 사업자, 다양한 서비스 제공업자 들이 모여 이루어지는 모바일과 유비쿼터스 컴퓨팅에서는 이러한 본질적 가정이 어려울 수밖에 없다. 즉, 특정 영역에서의 다수의 이질적인 온톨로지가 만들어지고 모바일 디바이스, 서비스 제공업자들은 서로 다른 온톨로지를 사용한다. 따라서 "의미적 상호 운용성"을 위한 해결 방안이 요구된다. 의미적 상호 운용성(semantic interoperability)이란 유사한 영역에서의 같은 개념을 다른 어휘로 표현하는 다중 도메인 사이에서의 정확한 의미 전달을 보장하는 문제이다[6].

따라서 본 논문에서는 모바일과 유비쿼터스 컴퓨팅에서의 의미적 상호 운영성의 문제를 해결하기 위한 온톨로지 변환 시스템을 소개한다. 특별히 제안하는 변환 시스템의 특징은 변환 규칙의 자동 생성이 연쇄적으로 이루어지기 때문에 변환 규칙 작성이 쉽다는 것이다. OntoMorph[7] 또한 기존의 연구되어진 온톨로지 변환 시스템인데, 하지만 A 도메인의 온톨로지 데이타를 B 도메인의 온톨로지 데이타로 바꾸는 변환 규칙을 관리자가 항상 수작업으로 직접 작성해야 한다(이후부터는 이러한 수작업에 의해 변환 규칙을 직접 작성하는 것을 "직접 변환 규칙 작성"이라고 함). 따라서 최악의 경우 N개의 도메인이 존재할 경우 $O(N^2)$의 변환 규칙 작성 복잡도가 존재한다. 비록 제안하는 방법 또한 $O(N^2)$의 변환 규칙 작성 복잡도를 갖지만, 연쇄성에 의하여 이를 실제적으로 개선할 수 있다. 향후 다양한 모바일과 유비쿼터스 디바이스들이 자신의 온톨로지를 사용하고, 또한 이러한 온톨로지들이 갱신되어 다양한 온톨로지 버전이 생성됨을 가정할 때, 이러한 변환 규칙 작성의 복잡도를 개선하는 것은 관리적 측면에서 중요하게 여겨질 것이다.

본 논문에서는 연쇄 조합에 의한 변환 규칙 작성의 적용을 일차적으로 RDF 메시지 데이타에 적용한 연구 결과를 소개한다. RDF 보다 더 풍부한 온톨로지 기능을 제공하며 더 복잡한 표현 어휘를 갖는 OWL 온톨로지 데이타에 대한 변환 시스템의 연구는 추후 과제이다. RDF는 원래 웹 자원들에 대한 메타 데이타를 기술하기 위한 표준 프레임워크로 개발되었지만, 더 발전하여 지금은 웹 온톨로지 표준 언어인 OWL의 기반이 되었기 때문에 본 논문에서는 RDF 데이타 또한 작은 의미에서의 온톨로지 데이타로 간주한다.

본 논문의 구성은 다음과 같다. 먼저 2장에서는 온톨로지 상호 운용성의 문제를 연구한 기존의 주요 연구들을 살펴본다. 3장에서는 실시간 온톨로지 변환 시스템의 개념 및 본 연구의 기본 가정과 응용 환경을 정의한다. 4장에서는 본 논문이 해결하고자 하는 문제점 및 연쇄 조합 변환 규칙 생성의 개념을 소개하며, 5장에서는 이를 XML 데이타 모델과 XQuery 중첩 질의의 특성을 이용하여 구현한 방법을 소개한다. 6장에서는 제안된 방법을 실험을 통하여 분석하며, 또한 2장에서의 기존 연구와의 비교 고찰을 언급한다. 7장에서는 본 연구의 결론을 맺는다.

2. 관련 연구

이질적인 온톨로지들 사이에서의 의미적 상호 운용성을 다룬 주요 선행 연구들은 크게 병합(merging), 변환(translation), 맵핑(mapping)에 의한 방법으로 분류될 수 있다. 본 연구는 기본적으로 변환에 의한 방법인데, 이는 변환이 RDF 메시지와 같은 데이타(즉, 질의나 규칙이 아닌)의 의미적 상호 운용성에 가장 실용적인 방법이라고 생각하기 때문이다.

- 두 온톨로지를 병합 (merging)하여 새로운 단일 온톨로지를 구축하는 방법

OntoMerge[6]는 점진적인 병합 방법을 사용한다. 즉 먼저 두 지역 온톨로지(local ontology)를 병합하여 초기 전역 온톨로지(global ontology)를 생성하고, 새로운 지역 온톨로지가 생길 때 마다 이를 전역 온톨로지에 병합하는 방법을 사용한다. 여기서 병합은 두 지역 온톨로지를 대표할 수 있는 새로운 어휘를 만들고 각 지역 온톨로지의 어휘로부터 전역 온톨로지의 어휘로의 관계를 정의하는 규칙을 만드는 것을 의미한다. 이러한 규칙을 브리징 공리(bridging axioms)라고 하는데, 전역 온톨로지에 같이 포함된다. 이는 전역 온톨로지가 클래스 및 속성과 브리징 공리에 대하여 일차 술어 논리 언어(first-order logic language) 형식으로 기술되기 때문이다. 따라서 고전적인 추론(OntoMerge는 인덱싱에 의한 보다 최적화된 추론 방식을 사용)에 의해, 소스 온톨로지의 데이타를 타겟 온톨로지 데이타로 변환할 수 있다. OntoMerge는, 지역 온톨로지들 사이의 각 쌍에 대해 병합 작업을 모두 수행하는 단순한 방법과 비교할 때 (이는 $O(N^2)$의 병합 작업을 요구), 전역 온톨로지와의 병합 $O(N)$만을 요구한다.

Chimaera[8]와 IPROMPT[9]는 사용자가 수작업으로 병합을 수행하는 것을 도와 줄 수 있는 자동 병합 기능, 병합 작업이 수행될 필요성이 있는 부분의 제안 기능, 데이타의 불일치성, 문제점 등을 확인하는 기능 등의 준자동적인(semi-automatic) 사용자 인터페이스의 개념을 소개한다.

- 하나의 온톨로지 인스턴스 데이타를 다른 온톨로지 인스턴스 데이타로 변환(translation) 하는 변환 규칙 작성

Ontolingua[10]는 OntoMerge와 유사하게 상이한 지역 온톨로지 사이의 연결 고리 역할을 수행하는 전역 온톨로지가 존재함을 가정한다. 각 지역 온톨로지는 전역 온톨로지에 대한 변환 규칙을 갖게 된다. 따라서 전역 온톨로지와의 $O(N)$의 변환 규칙 작성 복잡도가 존재한다. 하지만 모든 지역 온톨로지 관리자들이 동의하는 전역 온톨로지 구축에 대한 비현실성과, 이의 유지보수에 대한 어려움이 존재한다.

OntoMorph[7]는 Ontolingua와 달리 지역 온톨로지들 상호 간에 직접적으로 작성된 직접 변환 규칙을 통하여 서로의 의미 정보를 해석한다. 하지만 최악의 경우 $O(N^2)$의 변환 규칙 작성 복잡도가 존재한다.

본 연구에서는 OntoMorph와 같이 전역 온톨로지를 배제한 지역 온톨로지들 상호 간의 변환 방법을 연구하며, 하지만 연쇄성에 의하여 $O(N^2)$의 변환 규칙 작성 복잡도를 개선하고자 한다.

- 두 온톨로지 사이의 맵핑 (mapping) 관계를 정의

이 방법은 두 지역 온톨로지 사이에서의 동일 개념에 대하여 맵핑 규칙을 작성하는 것, 혹은 하나의 전역 온톨로지와 지역 온톨로지 사이의 동일 개념에 대하여 맵핑 규칙을 작성하는 것으로, OBSERBER[11], ONION [12], *GLUE*[13] 등의 연구가 있다. *OBSERVER*의 IRM (Interontology Relationship Manager)은 별도의 독립적인 서버에 지역 온톨로지들 사이의 동일한 의미를 갖는 어휘 사이의 동의어 관계(synonym)만을 저장한다. GLUE는 두 온톨로지 사이의 맵핑되는 개념을 찾기 위해, 온톨로지 인스턴스 데이타와 온톨로지 분류 체계(taxonomy) 정보를 입력으로 하여 기계학습 알고리즘을 수행한다.

이러한 온톨로지 의미적 상호 운용성을 다룬 기존의 연구들은 공통적으로 온톨로지 이질성을 두 가지로 분류하여 해결하고자 하였다. 하나는 온톨로지 언어의 차이로 인한 구문적 이질성(syntactic heterogeneity)과 다른 하나는 온톨로지 어휘 및 분류 체계의 디자인 차이로 인한 의미적 이질성(semantic heterogeneity)이다. 본 연구에서는 RDF, OWL과 같은 웹 온톨로지 표준 언어가 이미 정의되고 있는 현 시점에서 구문적 이질성은 배제하기로 하고, 의미적 이질성만을 고려하기로 한다.

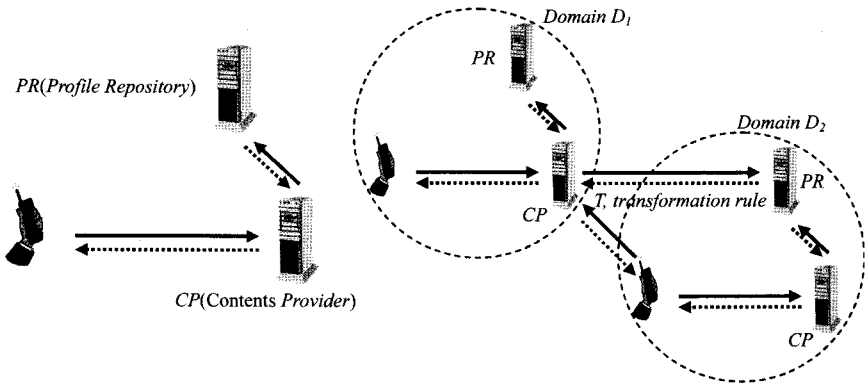
3. 실시간 온톨로지 변환 시스템의 소개 및 기본 가정

3.1 단순(simple) CPI 메시지 데이타를 위한 온톨로지 변환 시스템

먼저 온톨로지 변환 시스템의 정의는 소스(source) 온톨로지 인스턴스 데이타를 미리 정의된 변환 규칙(translation rule)에 의하여 타겟(target) 온톨로지 인스턴스 데이타로 변환시키는 것을 말한다.

정의 3.1 $O_s \xrightarrow{T} O_t$, O_s, O_s: 소스 온톨로지 인스턴스 데이타, O_t: 타겟 온톨로지 인스턴스 데이타, T: 변환 규칙

그림 1(b)는 하나의 예로 모바일 컴퓨팅에서 응용될 수 있는 온톨로지 변환 시스템의 개념을 소개한다. 먼저 현재 표준인 그림 1(a)의 OMA UAProf(User Agent



(a) 단일 도메인 (b) 다중 도메인

그림 1 모바일 컴퓨팅에서의 실시간 온톨로지 변환 시스템의 개념

Profile)[3] 혹은 W3C의 CC/PP(Composite Capabilities/ Preferences Profile)[14]의 사용 시나리오를 살펴보자. UAProf와 CC/PP는 디바이스의 지원 사양(capabilities), 운영 환경, 네트워크 환경, 그리고 콘텐츠 사용에 관한 사용자의 관심을 기술할 수 있는 CPI(Capabilities and Preference Information)를 정의한다. UAProf와 CC/PP는 RDF 온톨로지 언어로 기술된다. 이러한 CPI 데이타에 따라 콘텐츠 제공자(CP, Contents Provider) 혹은 네트워크 구성 요소들은 디바이스에 적합한 서비스와 통신을 제공할 수 있다. 그림 1(a)의 클라이언트 디바이스는 자신의 디바이스 프로파일정보(CPI)를 가지고 있는 프로파일 저장소(PR)의 URI를 콘텐츠 서버(CP)에 알려준다. 콘텐츠 서버는 디바이스 프로파일 정보를 요청하고, 프로파일 정보에 따라 디바이스에 적

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#" xmlns:prf1="mobile1_schema1#">
   <rdf:Description rdf:ID="Device1Profile">
      <prf1:component>
         <prf1:HardwarePlatform>
            <prf1:BluetoothProfile>
               <rdf:Bag>
                  <rdf:li>headset</rdf:li>
                  <rdf:li>dialup</rdf:li>
                  <rdf:li>lanaccess</rdf:li>
               </rdf:Bag>
            </prf1:BluetoothProfile>
            <prf1:ScreenSize>121*87</prf1:ScreenSize>
            <prf1:Model>R999</prf1:Model>
            <prf1:BitsPerPixel>2</prf1:BitsPerPixel>
            <prf1:ColorCapable>NO</prf1:ColorCapable>
            <prf1:ImageCapable>NO</prf1:ImageCapable>
                  : :          : :          : :
         </prf1:HardwarePlatform>
      </prf1:component>
      <prf1:component>
         <prf1:SoftwarePlatform>
            <prf1:CcppAccept>      <!-- List of content types the device supports -->
               <rdf:Bag>
                  <rdf:li>application/vnd.wap.multipart.mixed</rdf:li>
                  <rdf:li>text/vnd.wap.wml</rdf:li>
                  <rdf:li>text/vnd.wap.wmlscript</rdf:li>
                  <rdf:li>text/x-vCard</rdf:li>
                  <rdf:li>text/x-vCalendar</rdf:li>
                  <rdf:li>text/x-vMel</rdf:li>
                  <rdf:li>text/x-eMelody</rdf:li>
                  <rdf:li>image/vnd.wap.wbmp</rdf:li>
                  <rdf:li>image/gif</rdf:li>
               </rdf:Bag>
            </prf1:CcppAccept>
                  : :          : :          : :
         </prf1:SoftwarePlatform>
               : :          : :          : :
      </prf1:component>
   </rdf:Description>
</rdf:RDF>
```

그림 2 mobile1 통신 회사의 Device1에 대한 CPI 데이타

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#" xmlns:prf2="mobile2_schema1#">
  <rdf:Description rdf:ID="Device2Profile">
    <prf2:component>
      <prf2:HW>
        <prf2:BT>
          <prf2:ModelInfo>BT2006</prf2:ModelInfo>
          <prf2:SupportedFunction>
            <rdf:Bag>                          <!-- code  function_name -->
              <rdf:li>3090</rdf:li>            <!-- 3090  'headset' -->
              <rdf:li>3355</rdf:li>            <!-- 3355  'lanaccess' -->
            </rdf:Bag>
          </prf2:SupportedFunction>
        </prf2:BT>
        <prf2:Screen>
          <prf2:Width>12</prf2:Width>
          <prf2:Height>87</prf2:Height>
          <prf2:Color>4</prf2:Color>
          <prf2:ColorCapable>YES</prf2:ColorCapable>
        </prf2:Screen>
          : :          : :          : :
      </prf2:HW>
    </prf2:component>
    <prf2:component>
      <prf2:SW>
        <prf2:CcppAccept>          <!-- List of content types the device supports -->
          <rdf:Bag>
            <rdf:li>application/vnd.wap.multipart.mixed</rdf:li>
            <rdf:li>text/vnd.wap.wml</rdf:li>
            <rdf:li>text/vnd.wap.wmlscript</rdf:li>
            <rdf:li>text/x-vCard</rdf:li>
            <rdf:li>text/x-vCalendar</rdf:li>
            <rdf:li>text/x-vMel</rdf:li>
            <rdf:li>text/x-eMelody</rdf:li>
            <rdf:li>image/vnd.wap.wbmp</rdf:li>
            <rdf:li>image/gif</rdf:li>
            </rdf:Bag>
        </prf2:CcppAccept>
          : :          : :          : :
      </prf2:SW>
    </prf2:component>
          : :          : :          : :
  </rdf:Description>
</rdf:RDF>
```

그림 3 mobile2 통신 회사의 Device2에 대한 CPI 데이타

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#" xmlns:prf3="mobile3_schema1#">
  <rdf:Description rdf:ID="Device3Profile">
    <prf3:component>
      <prf3:Hardware>
        <prf3:BluetoothHeadset>YES</prf3:BluetoothHeadset>
        <prf3:Bluetoothlanaccess>YES</prf3:Bluetoothlanaccess>
        <prf3:LCDSize>121*87</prf3:LCDSize>
        <prf3:ColorPalette>2</prf3:ColorPalette>
        <prf3:ColorSupport>YES</prf3:ColorSupport>
          : :          : :          : :
      </prf3:Hardware>
    </prf3:component>
    <prf3:component>
      <prf3:Software>
        <prf3:CcppAccept>          <!-- List of content types the device supports -->
          <rdf:Bag>
            <rdf:li>application/vnd.wap.multipart.mixed</rdf:li>
            <rdf:li>text/vnd.wap.wml</rdf:li>
            <rdf:li>text/vnd.wap.wmlscript</rdf:li>
            <rdf:li>text/x-vCard</rdf:li>
            <rdf:li>text/x-vCalendar</rdf:li>
            <rdf:li>text/x-vMel</rdf:li>
            <rdf:li>text/x-eMelody</rdf:li>
            <rdf:li>image/vnd.wap.wbmp</rdf:li>
            <rdf:li>image/gif</rdf:li>
            <rdf:li>audio/vnd.mobile3.poly;ver=4x</rdf:li>
          </rdf:Bag>
        </prf3:CcppAccept>
          : :          : :          : :
      </prf3:Software>
    </prf3:component>
          : :          : :          : :
  </rdf:Description>
</rdf:RDF>
```

그림 4 mobile3 통신 회사의 Device3에 대한 CPI 데이타

합한 콘텐츠로 변형하여 제공한다. 예로, 그림 2는 하드웨어 특성(<HardwarePlatform>)과 소프트웨어 특성(<SoftwarePlatform>)에 관한 UAProf CPI 데이타의 일부를 보여 주는데, <ScreenSize> 단말 노드(leaf node)의 값이 '121*87'이므로 CP는 이러한 디바이스 화면 크기에 맞도록 콘텐츠를 변형하여 제공한다.

비록 UAProf와 CC/PP 명세가 표준이기 때문에 이를 따르는 통신회사, CP, 모바일 디바이스 사이의 상호 운용성이 보장되지만, 그림 1(b)와 같이 다른 어휘와 분류체계를 사용하는, 그리고 RDF 문서의 단말 노드가 갖는 값의 단위, 표현 등이 통신 회사마다 다를 수 있다. 즉, 그림 2, 3, 4는 각각 유사한 영역에서의 mobile1, mobile2, mobile3 통신 회사의 Device1, Device2, Device3에 관한 이질적인 CPI 온토롤지 인스턴스 데이타를 보여준다.

정의 3.2 단일 도메인: 동일 온톨로지를 사용, 다중 도메인: 유사한 영역에서의 서로 상이한 온톨로지를 사용하는 단일 도메인들의 집합

그림 1(b)의 다중 도메인에서의 온톨로지 변환 시스템의 사용 시나리오는 다음과 같다. 먼저 D_2의 PR은 사전에 D_2의 CPI를 D_1의 CPI로 변환할 수 있는 변환 규칙을 작성해 두었다고 가정하자. 도메인 D_2의 클라이언트 디바이스는 도메인 D_1의 CP에 자신의 프로파일 정보를 가지고 있는 PR의 URI를 알려준다. 도메인 D_1의 CP는 자신이 사용하고 있는 CPI 온톨로지에 관한 정보를 도메인 D_2의 PR에 알려주고, D_2의 PR로부터 디바이스 프로파일 정보 및 D_2의 CPI를 D_1의 CPI로 변환할 수 있는 변환 규칙을 전송 받는다. D_1의 CP는 변환 규칙을 사용하여 자신이 이해할 수 있는 디바이스 프로파일 정보로 변환한 후, 프로파일 정보를 사용하여 디바이스에 적합한 콘텐츠로 변형하여 서비스한다.

3.2 가정 및 응용 환경

본 연구에서는 다음의 중요한 몇 가지 가정 및 응용 환경을 전제로 한다.

가정 1. 온톨로지 변환의 대상은 데이타뿐만 아니라, 질의(queries), 규칙(rules)등 다양할 수 있다[6]. 하지만 본 연구의 초점은 모바일과 유비퀴터스 컴퓨팅에서의 UAProf, CC/PP와 같은 RDF 언어로 표현된 간단한 메시지 데이타(예로, 상황인식 시스템의 상황 정보 메시지, 센서 시스템의 센서 정보 메시지, 로봇 사이의 교환 메시지, 인터넷 응용 메시지 등)의 변환이다. "간단한(simple)"의 의미는 이러한 응용 환경에서 메시지 데이타의 특성은 그 크기가 작고, 스키마가 복잡하지 않을 것이라는 것을 의미한다.

가정 2. 모바일과 유비퀴터스 컴퓨팅에서의 간단한 RDF 메시지 데이타는 반드시 하나의 목적 개념에 대한 기술(description) 정보를 가지는 것을 가정한다. 예로, UAProf, CC/PP 데이타는 하나의 디바이스(디바이스 프로파일 자체가 목적 개념)에 대한 특성을 기술하며, <HardwarePlatform>, <SoftwarePlatform> 등의 개념은 목적 개념의 특성을 서술하기 위한 개념으로 사용되고 있다.

가정 3. 이러한 RDF 메시지 데이타의 스키마는 반복되는 개념이 표현될 가능성이 적도록 디자인 되는 것을 가정한다. 또한 개념이 반복될 경우, 반복의 횟수가 적고, 반복 개념의 서술 내용의 크기가 작다는 것을 가정한다.

예 3.1 UAProf[3] 메시지의 경우 특정 디바이스에 대한 <HardwarePlatform>, <SoftwarePlatform>, <BrowserUA>, <NetworkCharacteristics>, <WapCharacteristics>, <PushCharacteristics>에 대한 개념을 한 번씩만 기술한다. 유일하게 반복되는 것은 그림 2의 <CcppAccept> 태그의 <Bag> 컨테이너 태그와 같이 반복되는 문자열 (literals) 값을 갖는 경우이다.

예 3.2 아래의 경우(참고문헌 [4]의 예제 데이타) <person> 개념은 반복되지만 횟수가 적고 그 내용은 <name>으로만 이루어져 크기가 작다.

```
<pol:creator>
  <per:Person>
    <foaf:name>Tim Finin</foaf:name>
  </per:Person>
  <per:Person>
    <foaf:name>Harry Chen</foaf:name>
  <per:Person>
</pol:creator>
```

가정 4. 각 도메인에서의 변환 규칙은 웹을 통하여 여러 도메인에 개방되어 제공될 수 있음을 가정한다.

4. 문제 정의 및 연쇄 조합 변환 규칙 생성

본 연구에서의 온톨로지 상호 운용성을 위한 기본적인 접근 방법은 2장의 관련 연구에서 소개한 Onto-Morph의 지역 온톨로지들 사이에서의 직접 변환 규칙 작성(direct composition of a translation rule)을 기반으로 한다. 이는 지금까지 온토롤지 상호 운용성을 연구한 다수의 연구에서 언급하듯이, 단 하나의 공유되는 전역 온톨로지를 구축하는 것은 매우 어려운 것이라고 판단하기 때문이다[7]. 또한 여러 관련 전문가들 또한 이러한 생각을 공유한다. 따라서 서로 상이한 온톨로지를 사용하는 다중 도메인 사이에서는 서로에게 적합한 변환 규칙을 정의하여 운영하는 것이 가장 쉬우면서도 단순한 방법이라고 생각한다. 하지만, 이러한 방식은, N개의 지역 온톨로지를 가정할 경우, 최악의 경우 $O(N^2)$의 변환 규칙 작성 복잡도를 가지고 있다. 본 연구에서는

현재 웹의 개방성(가정 4)이라는 추구 논리와 연쇄 조합 변환 규칙 생성(cascade composition of translation rules)의 개념을 통하여 이러한 변환 복잡도를 개선해 보고자 한다.

• 연쇄 조합 변환 규칙 생성의 개념

그림 5(a)의 방향성 그래프(directed graph)를 고려하자. 그래프의 각 노드는 유사한 개념을 정의하는 상이한 온톨로지 도메인 D_1, D_2, D_3, D_4, D_5를 나타내고, 두 노드 사이의 연결선은 온톨로지 직접 변환 규칙 작성을 나타낸다. 예로, D_2에서의 D_1으로의 연결선은 D_2의 온톨로지 인스턴스 데이타를 D_1으로 변환하는 변환 규칙을 D_2가 작성하였음을 나타낸다. 만약 D_2에서 D_1으로의 변환 규칙을 D_3에 제공할 수 있음을 가정하면(가정 4), D_3에서는 D_3 -> D_2 변환 규칙을 통하여 변환된 D_2 온톨로지 인스턴스 데이타를 다시 D_2 -> D_1 변환 규칙을 통하여 D_1 온톨로지 인스턴스 데이타로 자동 변환시킬 수 있다. 마찬가지로 D_5의 D_3로의 변환 규칙이 작성되고 D_3 -> D_2, D_2 -> D_1 변환 규칙이 D_5에 제공되면, 일련의 변환 과정 D_5 -> D_3, D_3 -> D_2, D_2 -> D_1을 통하여 D_5 온톨로지 인스턴스 데이타를 D_1 온톨로지 인스턴스 데이타로 자동 변환시킬 수 있다. 하지만 이러한 연쇄적인 변환 방식은 중간 변환 결과물이 매 변환 과정마다 생성되어야 하므로 여러 번의 변환 규칙이 적용되는 경우 성능상 비효율적이다. 따라서 변환 규칙을 단계적으로 조합하는 방법(cascade composition)을 고려한다. 즉, D_3 -> D_2, D_2 -> D_1의 변환 규칙이 존재한다면 D_3 -> D_1의 변환 규칙이 자동으로 조합 생성될 수 있는 가능성을 가진다. 또한 D_3 -> D_2의 직접 변환 규칙과 D_3 -> D_1의 연쇄 조합 변환 규칙을 D_5가 제공받는다면, D_5 -> D_3에 의해 D_5 -> D_2와 D_5 -> D_1의 연쇄 조합 변환 규칙을 자동으로 생성할 수 있는 가능성을 가진다. 이러한 연쇄 조합 변환 규칙이 최적으로 작성될 수 있음을 가정하면, 연쇄 조합 변환 규칙에 의한 한 번의 변환은 일련의 연쇄적인 변환 방식보다는 빠른 변환 시간을 보장할 것이다.

하지만 연쇄 조합 변환 규칙 생성은, 만약 그러한 연쇄적인 변환 규칙 조합이 일어나지 않는 경우를 고려한다면, 여전히 최악의 경우(in the worst case) $O(N^2)$의 변환 규칙 작성 복잡도를 가지게 된다.

정리 4.1 연쇄 조합 변환 규칙 생성의 최악의 경우 변환 규칙 작성 복잡도는 $O(N^2)$이다.

증명. 일련의 상이한 온톨로지 도메인 D_1, D_2, D_3, ..., D_N을 고려하자. 먼저 D_1으로부터의 나머지 온톨로지 도메인들로의 직접 변환 규칙 작성을 고려하면, 변환 규칙 작성의 횟수는 $(N-1)$개이다. 다음으로, D_2의 나머지 온톨로지 도메인들로의 직접 변환 규칙 작성을 고려하자. 여기서 D_2 -> D_1은 배제한다. 왜냐하면 D_1 변환 규칙과의 연쇄 조합에 의하여 나머지 온톨로지 도메인들로의 변환 규칙을 자동으로 얻을 수 있기 때문이다. 따라서 D_2의 변환 규칙 작성의 횟수 $(N-2)$개를 고려한다. 또한 D_3의 경우, D_3 -> D_1과 D_3 -> D_2를 배제하므로, 변환 규칙 작성의 횟수는 $(N-3)$개이다. 이와 같이 연쇄 조합 변환 규칙 생성이 일어나지 않는 변환 규칙 작성을 고려하면, 어떤 온톨로지 도메인 D_k의 경우 변환 규칙 작성 횟수는 $(N-k)$개이다, $1 \le k < N$. 따라서 변환 규칙 작성의 총 횟수는 $\sum k$, $1 \le k < N$이다. 그림 5(b)는 이러한 예를 보여준다. □

비록 연쇄 조합 변환 규칙 생성이 근본적으로 $O(N^2)$의 복잡도를 갖지만, 본 연구에서는 변환 규칙의 연쇄적인 조합이 일어날 수 있는 가능성이 높다고 확신한다. 따라서 상기 개념을 통한 변환 규칙 작성 복잡도의 개선의 효과를 기대할 수 있다. 또한 연쇄 조합 변환 규칙 생성은 최선의 경우(in the best case) 변환 규칙 작성 복잡도가 $O(N)$이다. 그림 5(c)는 이러한 예를 보여준다.

다음 장에서는 이러한 연쇄 조합 방법에 의한 변환 규칙의 자동 생성의 가능성을 XML 데이타의 특성과 XQuery의 중첩 질의의 특성을 이용하여 이루어질 수 있음을 보인다.

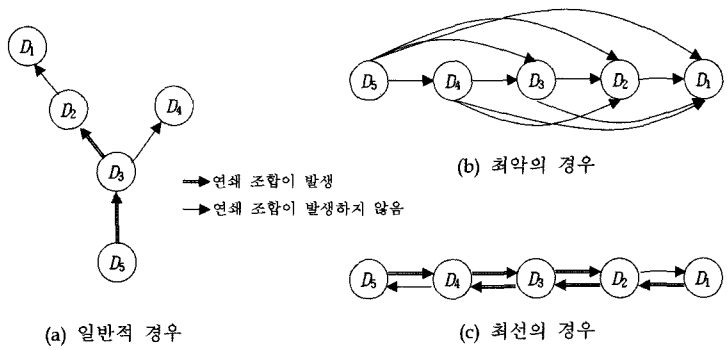


그림 5 연쇄 조합 변환 규칙 생성

5. 제안 방법

5.1 변환 규칙 형식

먼저 시스템에서 정의하는 변환 규칙의 형식은 다음과 같다.

(1) 변환 규칙은 RDF 타겟 온톨로지 인스턴스 데이타와 동등한 태그 이름 및 구조를 갖는다. 예로, 부록 1의 mobile2의 CPI 데이타를 mobile1의 CPI 데이타로 변환하기 위한 변환 규칙은 그림 2의 mobile1의 태그 이름과 구조와 동일함을 함을 알 수 있다.

(2) 변환 규칙에 사용되는 개념들은 타겟 온톨로지와 매칭되는 개념 중 최하위 개념으로 표현된다. 이는 최하위 개념은 상위 개념으로 해석될 수 있기 때문이다. 예로, 타겟 온톨로지에서 <publication>이 <article>의 상위 개념이고 소스 온톨로지의 <journalarticle>이 <publication>과 <article> 두 개념에 모두 매칭될 경우, 변환 규칙은 타겟 온톨로지의 <article>에 대하여 기술된다.

(3) 변환 규칙의 단말 노드는 소스 온톨로지 인스턴스 데이타 값을 이용하여 타겟 온톨로지에 적합한 데이타 값으로 변환하기 위한 단 하나의 XQuery 질의 문장을 갖는다. 예로, 부록 1의 T2.xq의 모든 단말 노드는 단 하나의 XQuery 질의 문장을 갖는다.

5.2 연쇄 조합 변환 규칙 생성

변환 규칙 또한 온톨로지 인스턴스 데이타와 동일한 태그 구조를 갖는 다는 것과 일련의 변환 규칙들의 XQuery를 중첩시킬 수 있다는 것에 의해, 연쇄 조합 변환 규칙의 생성이 자동화 될 수 있다. 생성 알고리즘은 그림 6과 같다. 첫 번째 인자인 $r1$은 4장의 "연쇄 조합 변환 규칙 생성의 개념"에서 설명한 D_3가 제공하는 D_3 -> D_1과 같은 임의의 연쇄 조합 변환 규칙 혹은 D_3 -> D_4와 같은 직접 변환 규칙을 인자 값으로 가지며, $r2$는 D_5가 D_3에 대한 작성한 D_5 -> D_3와 같은 직접 변환 규칙을 인자 값으로 갖는다. 이때 *Casecade-Composition* 함수는 D_5 -> D_1 혹은 D_5 -> D_4의 연쇄 조합 변환 규칙을 생성 반환한다.

위의 알고리즘 수행 시의 몇 가지 특별한 연쇄 조합의 경우를 살펴보면 다음과 같다.

예 5.1 부록 1의 T2.xq 변환 규칙은 그림 2의 mobile1의 온톨로지 인스턴스 메시지와 동일한 태그 구조를 갖는다. 또한 부록 2의 T3.xq 변환 규칙은 그림 3의 mobile2의 온톨로지 인스턴스 메시지와 동일한 태그 구조를 갖는다. 따라서 T2.xq의 <ScreenSize> 단말 노드의 값인 XQuery 질의문의 XPath는 T3.xq의 <Width>와 <Height>의 값을 참조할 수 있다. 위 알고리즘 단계 2-i)ii)에 의하여, T3.xq의 <Width>와 <Height>의 XQuery 질의문은 부록 3의 T4.xq의 <ScreenSize>와 같이 중첩된다.

예 5.2 알고리즘 단계 2-iii)에 의하여, 부록 3 T4.xq의 CcppAccept XQuery 질의의 $a, $b, $c와 같이, 중첩되는 경우 변수명이 자동으로 조정된다. 또한 <ScreenSize>의 중첩질의는 원래는 다음과 같으나, 참조되지 않는 변수와 관련된 의미 없는 질의 문장이 제거되었다.

```
let $a := doc("test2.rdf")
return
   concat(concat(
      let $b := doc("test3.rdf")
      return
substring-before($b/rdf:RDF/rdf:Description/prf3:co
mponent/rdf:Description[@rdf:ID="Hardware"]/prf3:
LCDSize/text(), "*"), "*"),
      let $b := doc("test3.rdf")
      return
substring-after($a/rdf:RDF/rdf:Des-
cription/prf3:component/rdf:Description[@rdf:ID="H
```

Rule CascadeComposition (Rule r1, Rule r2)

/* $r1$은 연쇄 조합 변환 규칙 혹은 직접 변환 규칙, $r2$는 직접 변환 규칙 */

1. 변환 규칙 $r1$의 *declare* 부분을 제외한 <rdf:RDF> 태그 이하를 DOM 파싱하여 $r1$의 DOM tree 생성
2. $r1$ DOM tree를 전위 순회 (preorder traverse)하며 만약 순회 노드가 단말 노드이며 노드 값이 "{" *XQuery statement* "}"인 경우 다음을 수행
   - i) XQuery에 표현된 모든 XPath에 대하여 $r2$에 질의
   - ii) 질의 결과에 해당하는 $r2$의 XQuery 문장을 $r1$의 XQuery에 중첩시킴
   - iii) 조합된 XQuery에서 변수 이름 조정 및 참조되지 않는 변수 제거
3. *Rule* $r3$ 생성
   - i) $r1$ DOM tree에서 사용되는 *declare statement*들을 $r1$, $r2$로부터 $r3$에 출력
   - ii) $r1$ DOM tree를 $r3$에 출력
4. *return* $r3$

그림 6 연쇄 조합 변환 규칙 생성 알고리즘

ardware"]/prf3:LCDSize/text(), "*"))

예 5.3 참고문헌 [15]는 XQuery 1.0에서 지원되는 함수들을 정의한다. 이외의 함수들은 T2.xq의 local:mk_mobile1_BluetoohProfile 함수와 같이 사용자 정의 함수(user-defined function, 변환 규칙이 함수 구현을 포함)형태로 제공되거나, log 함수와 같이 외부 함수(external function, 변환 규칙 외부에서 구현된 함수)로 제공된다[16]. 가정 4에서와 같이 외부 함수 또한 웹을 통하여 개방 공유될 수 있음을 가정한다. 이러한 declare statement들은 단계 3에 의하여 필요한 부분이 연쇄 조합 변환 규칙에 추가 된다.

예 5.4 가정 3의 반복 개념의 경우 직접 변환 규칙 작성 시 상위 태그까지 만을 기술한다. 예로 그림 2의 <BluetoothProfile>은 <rdf:Bag>의 반복 데이타를 갖는다. 따라서 부록 1 T2.xq는 <BluetoothProfile>까지 만의 XQuery 변환 질의를 갖는다. 역시 부록 2 T3.xq는 <SupportedFunction>까지 만의 변환 질의를 갖는다. 따라서 반복 개념 또한 부록 3 T4.xq의 <BluetoothProfile>과 같이 쉽게 중첩될 수 있다. 예 3.2의 경우도 추후 변환 규칙의 자동 조합을 지원하기 위해, 수작업에 의한 직접 변환 규칙 작성 시 <creator>까지 만의 변환 규칙 질의가 기술되어야 한다. 반복되어 지는 부분에 대한 데이타의 생성은 T2.xq의 local:mk_mobile1_BluetoothProfile 함수와 T3.xq의 local:mk_mobile2_SupportedFunction 함수와 같이 함수를 사용하여 처리한다.

예 5.5 부록 1의 변환 규칙 T2.xq의 <Model>과 같이 mobile2에 존재하지 않는 개념의 경우, 디폴트 상수 값을 설정하여 처리한다.

정리 5.1 이상과 같이 변환 규칙과 온톨로지 인스턴스 데이타가 동일한 태그 구조를 갖고 변환 규칙들의 XQuery가 중첩될 수 있다는 것에 의해, 연쇄 조합 변환 규칙의 생성이 자동화 될 수 있다.

증명. 일련의 RDF 문서 doc_1, doc_2, doc_3, ···, doc_n을 고려하자. doc_i를 doc_{i-1} (i = 2, 3, ..., n)로 변환하는 변환 규칙 T_i는 RDF 문서 doc_{i-1}과 같은 태그 구조를 갖는다. 이를 다음과 같이 표현하자: $tag\_pattern(T_i) \equiv tag\_pattern(doc_{i-1})$. 단, doc_{i-1}과는 달리, T_i는 부록 1 T2.xq에서의 어떤 태그 값의 생성을 위하여, doc_i로부터 관련되는 태그들의 값을 갖고 와서 doc_{i-1}의 태그 값으로 적합하도록 변환하는 일련의 서브 XQuery 질의들로 구성된다. 반복해서 T_{i-1}를 정의할 수 있고, $tag_pattern(T_i) \equiv tag_pattern(doc_{i-1})$이기 때문에 T_{i-1}는 T_i로부터 특정 태그에 해당되는 값을 가져 올 수 있다. 물론 이 값은 T_i의 XQuery 질의들이다. XQuery 1.0 [16]은 중첩 질의를 지원하므로 (부록 4에 EBNF로 표기된 XQuery 문법의 일부분을 첨부하였으며, Expr, ExprSingle에 의해 FLOWR문, 함수 호출 등이 중첩될 수 있음을 알 수 있다), 위의 예에서와 같이 T_i의 질의들은 T_{i-1} 질의들에 중첩될 수 있다. 이를 $T_{<i-1,\ i>}$로 표현하자. 따라서 doc_i는 변환 규칙 $T_{<i-1,\ i>}$에 의해 한번에 doc_{i-2}로 변환될 수 있다. 이와 같은 과정을 반복한다면 변환 규칙 $T_{<2,\ 3,\ ...,\ n>}$를 자동 생성할 수 있고, 이에 의해서 doc_i는 doc_1로 한 번에 변환될 수 있다. □

5.3 변환 규칙의 최적화를 위한 준 자동적(semi-automatic) 방법

자동적인 연쇄 조합 방법에 의하여 생성된 변환 규칙은 직접 변환 규칙을 작성한 경우보다 다소 많은 변환 시간이 걸릴 수 있다. 이것은 연쇄 조합에 의한 중첩 XQuery 질의가 직접 변환 규칙 작성에 비하여 의미 없는 질의 처리 부분을 가질 수 있기 때문이다. 따라서 연쇄 조합 질의의 최적화를 위한 준 자동적(semi-automatic) 연쇄 조합 방법을 소개한다. 준 자동적 연쇄 조합 방법은 먼저 앞서의 자동적인 연쇄 조합을 수행한 후 다음과 같은 몇 가지 경우에 대하여 관리자가 직접 변환 규칙을 작성하는 사용자 인터페이스를 제공하는 것이다.

(1) 연쇄 조합 XQuery 질의 문장의 크기가 임의의 임계값(threshold)을 넘어서는 지를 확인한다. 본 연구에서는 허용될 수 있는 연쇄 조합 변환 질의 문장의 크기 S를 다음과 같이 정의하고 임계값을 15로 정의하였다.

S = *#branch*($T.xq_i$) + *#depth*($T.xq_i$),

$T.xq_i$: 연쇄 조합 변환 규칙 T의 i번째 중첩 XQuery 질의,

#branch(q): 어떤 질의 q가 소스 온톨로지 인스턴스 데이타에서 참조하는 엘리먼트, 혹은 어트리뷰트의 수; 같은 엘리먼트를 참조하는 경우도 별도로 카운트한다.

#depth(q): 어떤 질의 q에 대한 소스 온토롤지와 타겟 온톨로지 사이의 도메인 수

예로, 부록 3의 T4.xq의 <ScreenSize>경우, 소스 온톨로지 인스턴스 데이타의 <LCDSize>를 두 번 참조하므로 *#branch*()의 값은 2 (중복 허용), mobile3 (소스 온톨로지) -> mobile2 -> mobile1 (타겟 온톨로지) 사이의 변환이므로 *#depth*()의 값은 3이다. S = 2 + 3은 임계값을 넘지 않으므로 이 부분은 자동 연쇄 조합 처리된다.

(2) R = *#branch*($T.xq_i$) / *#unique_branch*($T.xq_i$)의 값이 임의의 임계값을 넘어서는 지를 확인한다. 본 연구에서는 R을 7로 정의하였다. *#unique_branch*()는 *#branch*()와 마찬가지로 소스 온토롤지 데이타에서의 참조 엘리먼트 혹은 어트리뷰트의 수이지만, 중복을 허용하지 않는다. 예로, 부록 3의 T4.xq의

<ScreenSize>경우, 소스 온톨로지 인스턴스 데이타의 <LCDSize>를 두 번 참조하므로 *#unique_branch()*의 값은 1(중복 허용 않음)이고 R의 값은 2이다. R의 값이 높을수록 직접 변환 규칙을 작성을 할 경우 소스 온톨로지 인스턴스 데이타와 타겟 온톨로지 인스턴스 데이타가 1 : 1 혹은 1 : 2와 같은 적은 참조로 매칭될 높은 가능성을 가진다.

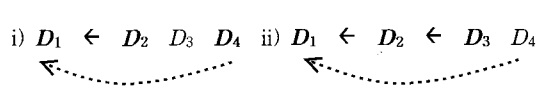
(3) 유사한 처리를 수행하는 하는 함수들이 연쇄 조합 XQuery 질의에 반복적으로 호출되는 지를 확인한다. 예로 부록 3의 T4.xq의 <ScreenSize> 경우, 문자열 처리 함수, concat, substring-before, substring-after가 반복적으로 호출되는데, 이러한 경우 더욱 간단한 매칭의 높은 가능성을 갖는다. 사실 그림 2의 <ScreenSize>의 값과 그림 4의 <LCDSize>의 값은 같다.

(4) 연쇄 조합 XQuery 질의에서의 함수 호출이 일정 패턴을 보이는 경우를 확인한다. 이러한 패턴으로는 아래의 경우와 같이 중간 기점을 중심으로 대칭 패턴을 보이는 부분이 있거나, 분기된 서브 쿼리의 두 패턴이 유사한 함수 호출 패턴을 보여 주는 부분이 있는 경우이다. 이러한 경우는 직접 변환의 경우 더욱 간단한 매칭의 높은 가능성을 갖는다.

```
upper-case(lower-case(upper-case(concat(substring-
before...
concat(lower-case(substring-before...,
        lower-case(substring-before...
```

(5) 변환 규칙 T의 전체 변환 시간에 비하여 xq_i들의 상대적인 변환 시간이 큰 질의들을 확인한다.

(6) 부록 1의 T2.xq의 <Model>은 문자열 상수 "R999 "의 디폴트 값을 갖는다. 이는 도메인 mobile1의 RDF 온톨로지 인스턴스 데이타는 <Model>를 정의하고 있지만, 도메인 mobile2는 이를 정의하고 있지 않기 때문이다. 따라서 참조의 깊이가 깊어지는 경우 다음과 같은 부정확한 매칭이 생길 수 있다. 즉 첫 번째 도메인 D_1, D_2, D_4에는 특정 개념이 존재하고, D_3에는 존재하지 않는 경우 D_3에서 D_2로의 변환 규칙 시 문자열 상수의 디폴트 값을 갖게 되고 이는 D_1까지 사용되는 값이 된다. 하지만 D_4에서 D_1으로의 직접 변환은 더욱 정확한 변환 규칙을 작성할 수 있다. 두 번째의 경우 물론 D_4는 특정 개념이 존재하지 않기 때문에 D_3에 대하여 디폴트 문자열 상수 값이 매칭될 수 있고 이 값이 계속 D_1으로 변환 규칙에 적용될 수 있지만, D_4에서 D_1으로의 직접 변환 규칙의 경우 더욱 정확한 디폴트 문자열 상수 값을 매칭할 수 있다. 이러한 확인은 변환 규칙의 단말 노드가 디폴트 상수 문자열 값을 갖거나 혹은 단말 노드의 XQuery 문장 내에 디폴트 상수 문자열이 사용되는 지를 확인하므로 수행된다.



특별히 규칙 (6)은, 규칙 (1)~(5)의 연쇄 조합 변환 규칙의 변환 시간을 직접 변환 규칙의 변환 시간에 근사하기 위한 것과는 달리, 연쇄 조합 변환 규칙의 적합성(soundness)과 관련된다. 적합성은 변환 규칙에 의한 왜곡된 변환(즉, 위의 예와 같이 소스 온톨로지 인스턴스 데이타에 관한 어떤 의미가 변환된 타겟 온톨로지 인스턴스 데이타에서 잘못 해석되거나 충분히 사용되지 못하는 경우)이 발생하지 않는 것을 의미한다.

비록 이러한 준 자동적인 인터페이스는 사용자의 직접적인 개입이 요구되지만, 실제적으로 자동적인 연쇄 조합 변환 규칙의 신속한 변환 규칙 작성의 이점을 크게 위배하지는 않는다고 생각한다. 왜냐하면 자동 조합 수행 후 몇몇 변환 시간이 크게 걸릴 곳을 자동으로 찾아내어 사용자에게 제시하고 또한 그 구조를 보여 줄 수 있기 때문에, 사용자는 빠른 시간에 쉽게 직접 변환 규칙의 작성을 수행할 수 있다. 또한 6.2절의 실제의 데이타를 통한 분석은, 유사한 영역에서의 이질적인 온톨로지 사이의 개념 매칭은 대부분 1:1의 높은 가능성을 가짐을 보여 준다.

6. 실험 및 분석

6.1 실험 환경

실험의 주된 분석은 제안된 연쇄 조합에 의한 변환 규칙의 변환 시간과 직접 작성된 변환 규칙의 변환 시간의 차이를 분석하는 것이다(6.2절과 6.3절). 또한 제안된 방법이 관련 연구에서 소개한 기존의 방법들과 어떠한 차별성을 갖는 지를 분석하는 것이다(6.4절).

6.2절과 6.3절에서는 변환 시간의 차이에 영향을 미치는 두 가지 요소를 기준으로 실험하였다. 표 1의 #branch는 5.3절에서 설명한 XQuery 변환 질의가 참조하는 엘리먼트 혹은 어트리뷰트의 수이다. #depth는 변환 질의가 몇 개의 도메인에 걸쳐 조합되었는지를 나타낸다.

실험 데이타로는, UAProf[3] 데이타와, 대학의 논문, 서적, 보고서 등의 출판 유형 분류에 대한 몇 가지 온톨

표 1 실험의 두 가지 요소

실험요소 (사용기호)	사용된 값
#branch	1, 2, 3, 4
#depth	3, 5, 7

로지 인스턴스 데이타[17-19]를 가공하여 사용하였다. 실험을 위하여서는 비슷한 주제를 다루는 하지만 상이한 의미적 이질성을 가지는, 다수의 온톨로지 인스턴스 데이타가 요구되는데, 이러한 다양한 데이타를 구하기가 어려웠다.

본 연구에서 주로 사용한 실험 도구는 Abacus Relational XQuery 질의 처리기[20]이다. Abacus XQuery 질의 처리기는 XQuery 1.0을 지원하며, 또한 JSR (Java Specification Requests) 225에서 규정하는 XQuery Java API(XQJ) 1.0[21]을 구현한다. 본 실험에서는 변환 질의 작성 및 변환 규칙의 변환 시간 측정을 위하여, Abacus XQJ API를 사용하여 Java 언어로 프로그래밍하였다. 변환 시간의 측정은 매 20회 시행하였으며, 대푯값으로 최빈값을 사용하였다.

실험은 Pentium(R) 듀얼 코어 프로세서 2.8 GHz CPU와 1 GB RAM을 가진 HP Pavilion 컴퓨터에서 수행되었다.

6.2 실험 인자 #branch와 #depth에 따른 분석

본 실험에서는 그림 2, 3, 4 이외 추가적으로 3개의 다른 의미적 이질성을 가지는 데이타를 인위적으로 만들어서 연쇄 조합에 의하여 만들어진 질의와 직접 변환에 의하여 만들어진 질의의 변환 시간의 차이를 분석하였다. 여기서 측정된 변환 시간은 변환 규칙 내의 아래의 #branch와 #depth의 특성을 갖는 하나의 특정 변환 질의의 변환 시간이다. 실험 RDF 메시지 데이타들의 크기는 약 10KB이다.

그림 7의 그래프에서 <1, 2, 1>의 1, 2, 1은 6.1절에서 설명한 #branch를 의미한다. 즉, 부록 1 T2.xq의 <ScreenSize>의 변환 질의는 mobile2의 두 엘리먼트를 참조하고, 부록 2의 T3.xq의 <Width>와 <Height>의 변환 질의는 각각 mobile3의 <LCDSize>를 참조하므로, 이를 연쇄 조합한 부록 3 T4.xq의 <ScreenSize> 변환 질의는 <1, 2, 1>의 특성을 갖는다. <1, 2, 1, 1, 2>은 #depth가 5인 경우이고, <1, 2, 1, 1, 2, 1, 1>은 #depth가 7인 경우이다.

그림 7의 그래프는 연쇄 조합 변환 질의(Cas)가 직접 변환 질의(Dir)보다 변환 수행 시간이 다소 큼을 보여준다. 이는 연쇄 조합 변환 질의는 도메인별 변환 질의를 조합한 경우이므로 클 수밖에 없다. 하지만 주요한 관찰은 #depth 5, 7 경우의 조합 변환 질의의 차이가 없다는 것이다. 이의 원인을 분석한 결과 비록 #depth가 일정 커졌지만, #branch의 크기는 변화가 없었기 때문이다.

그림 8의 경우는 #branch는 1의 크기로 유지하고, #depth를 달리한 경우인데, #depth가 7이 되는 경우 조합 변환 질의(Cas)의 시간에 변화가 있음을 확인할 수 있다.

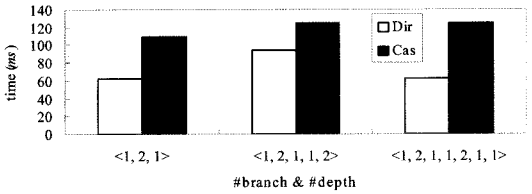


그림 7 직접 질의(Dir)와 연쇄 조합 질의(Cas)의 변환 시간 비교(#branch: 1, 2; #depth: 3, 5, 7)

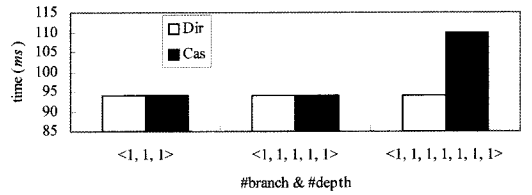


그림 8 직접 질의(Dir)와 연쇄 조합 질의(Cas)의 변환 시간 비교(#branch: 1; #depth: 3, 5, 7)

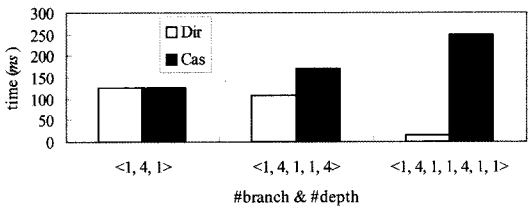


그림 9 직접 질의(Dir)와 연쇄 조합 질의(Cas)의 변환 시간 비교(#branch: 1, 4; #depth: 3, 5, 7)

그림 9의 <1, 4, 1, 1, 4, 1, 1>의 경우는 그림 7의 <1, 2, 1, 1, 2, 1, 1>와 비교하여 비록 #branch의 변화는 없었지만 #branch가 클 경우 #depth도 변환 시간에 영향을 주는 것을 확인할 수 있다. 본 실험을 통하여서, 작성된 연쇄 조합 변환 질의에 #branch와 #depth가 유의한 영향을 주는 것을 확인할 수 있었으며, #branch가 작은 경우에는 연쇄 조합 변환 질의와 직접 변환 질의의 변화 시간의 차이가 크지 않음을 확인할 수 있었다. 6.3절의 실제 데이타를 바탕으로 한 분석에서는 유사 도메인의 의미적 이질적 온톨로지 데이타 사이에서의 #branch의 수의 범위가 그렇게 크지 않음을 기대할 수 있다.

6.3 제안된 준 자동적 방법에 관한 실험

앞서의 실험은 UAProf[3] 데이타를 인위적으로 가공하여 만든 여러 의미적 이질성을 갖는 온톨로지 데이타에 대하여 변환 질의를 수행한 것이기 때문에 전체 변환 규칙의 변환 시간을 측정하는 것은 큰 의미가 없다고 판단하였다. 왜냐하면 전체 변환 규칙의 변환 시간은

단지 앞서의 실험에서 측정한 각 변환 질의의 합이기 때문이다. 따라서 실제 유사한 영역에서의 의미적 이질성을 갖는 몇 가지 온톨로지 인스턴스 데이타를 찾아보았다. 모바일과 유비쿼터스 컴퓨팅에 관련된 다양한 온톨로지 인스턴스 데이타를 구하는 것은 어려웠고, 본 실험에서는 DAML 웹 사이트의 3 가지의 university publication 온톨로지 인스턴스 데이타[17-19]를 이용하여 #depth 4의 연쇄 조합 변환 규칙의 변환 시간 측정 및 실제 데이타에 관한 연쇄 조합 변환 규칙의 의미를 고찰하여 보았다. 가정 2에 근거하여, 비록 university publication 온톨로지 인스턴스 데이타는 개념의 반복이 다수 존재할 수 있지만, 실험 데이타에서는 반복 개념이 존재하지 않도록 가공하여 사용하였다.

실제 데이타를 통한 중요한 실험 결과는 앞서 실험에서의 #branch의 크기가 대부분 유사 영역의 다양한 온톨로지들 사이에서 크지 않을 것이라는 것이다. 본 실험에서는 대부분 1:1 매칭이 이루어 졌고, 단지 값의 표현 형태가 다르기 때문에 이를 타겟 온톨로지에 적합하도록 변형하기 위한 함수를 사용하는데 추가 시간이 소모된 것뿐이었다. 이러한 추가 함수의 경우도 준 자동적 방법의 휴리스틱 규칙 3에 의하여 제거 되었고, 최종 연쇄 조합 변환 규칙은 직접 변환 규칙과 동일하게 되었다.

그림 10은 이러한 결과를 보여 준다. Cas는 자동 연쇄 조합 변환 규칙의 경우이고, Hcas는 준 자동 연쇄 조합 변환 규칙의 경우, Dir은 직접 변환 규칙 작성, CasDir은 각 도메인별 변환 규칙을 연속적으로 실행한 경우이다. 각 도메인별 개념이 대부분 일대일 매칭이 되기 때문에 Cas, Hcas, Dir이 큰 차이가 나지 않음을 확인할 수 있다. CasDir의 각 도메인의 변환 규칙을 연속적으로 적용하는 것은 예상대로 매우 좋지 않은 방법임을 확인할 수 있다. 이는 소스 온톨로지 데이타와 타겟 온톨로지 데이타 사이의 직접적인 변환 관계에서는 발생할 수 없는 전혀 의미 없는 데이타들이 중간 변환 단계에서는 발생하기 때문이다.

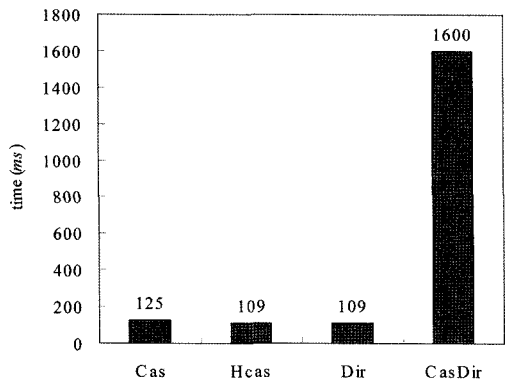


그림 10 변환 규칙의 총 수행 시간 비교

6.4 기존 방법과의 비교 분석

본 절에서는 앞서의 두 실험 및 연구 고찰을 통하여 제안된 방법이 가지는 의미를 기존 연구와의 비교를 통하여 살펴본다. 비교 기준은 표 2와 같다. 변환 시간은 변환 규칙에 의한 소스 온톨로지 인스턴스 데이타의 타겟 온톨로지 인스턴스 데이타로의 변환 시간이며, 변환 규칙의 적합성은 5.3절의 규칙 (6)에서 설명된 변환 규칙에 의한 왜곡된 변환이 존재하지 않는 것을 나타낸다. 변환 규칙 작성의 용이성은 규칙 작성 작업의 어려움 및 변화 규칙을 몇 번 작성해야 하는가에 대한 복잡도를 의미하며, 전역 온톨로지 구축은 4장 서두에서 언급하였듯이 전역 온톨로지 구축 및 유지 보수의 어려움을 나타낸다. 이전 온톨로지와의 상호 운용성은, 온톨로지를 갱신함으로써 이전 온톨로지와 새롭게 생성된 온톨로지 사이의 상호 운용성의 문제인데, 온톨로지 버전(versioning) 문제[22]와 관련된다. 마찬가지로 어플리케이션에 대한 영향은 온톨로지를 갱신함으로써 이를 사용하는 어플리케이션에 영향을 주는가의 문제이다. 비교 대상은 관련 연구에서 변환 방법으로 소개된 Ontolingua[10]와 OntoMorph[7]이다.

표 2의 분석에 의하면 두 가지 중요한 대립 관계를 갖는 특성은 (1) 변환 규칙을 작성하는 것을 용이하게 하는 것과 (2) 변환 규칙의 적합성과 빠른 변환 시간을 지원하는 것이다. OntoMorph는 수작업에 의해 작성된 직접 변환 규칙이기 때문에 최적의 변환 시간과 최적의

표 2 의미적 상호 운용성을 다룬 기존 연구와의 비교 분석

	변환 시간	변환 규칙의 적합성	변환 규칙 작성의 용이성	전역 온톨로지 구축	이전 온톨로지와의 상호 운용성	어플리케이션에 대한 영향
Ontolingua [10]	나쁨	좋음	- $O(N)$ 규칙 작성 복잡도 - 다소 어려움	필요	지원	없음
OntoMorph [7]	최적화	최적화	- $O(N^2)$ 규칙 작성 복잡도 - 어려움	불필요	지원	없음
제안된 준 자동적 방법	좋음	좋음	- $O(N^2)$ 규칙 작성 복잡도 - 쉬움, 연쇄성에 의한 복잡도 개선	불필요	지원	없음

적합성을 보장하지만, 최악의 경우 모든 도메인에 대하여 변환 규칙을 작성하여야 하기 때문에 변환 규칙의 작성, 유지 보수의 어려움이 존재한다.

Ontolingua의 경우는 기본적으로 추론을 통하여 타겟 온톨로지 인스턴스 데이타를 얻어 내는 것이므로 변환 시간이 다소 걸릴 것으로 판단된다. 또한 현실적으로 공유, 유지 보수되기 어려운 전역 온톨로지가 구축되어야 한다. 이러한 전역 온톨로지는 지역 온톨로지들 사이의 연결 관계를 중간에서 유지하는 것이므로, 지역 온톨로지들과 전역 온톨로지와의 직접 변환은 추후 지역 온톨로지와 지역 온톨로지 사이의 관계를 추론할 경우 좀 더 최적화된 변환을 놓칠 가능성이 있다. 전역 온톨로지는 그 크기가 계속 커지고 복잡해 질 것이므로, 이러한 전역 온톨로지와 지역 온톨로지의 연결 관계를 정의해 나가는 것은 점점 더 어려운 작업이 될 것이다.

제안된 준 자동적 방법은 4장에서 설명하였듯이, 비록 OntoMorph와 같이 기본적으로 $O(N^2)$ 규칙 작성 복잡도를 갖지만, 연쇄 조합에 의해 나머지 온톨로지들과의 변환 규칙을 자동적인 방법으로 얻을 수 있으므로, 보다 쉬운 변환 규칙 작성의 용이성을 갖는다. 또한 5.3절에서 소개되었듯이, 비록 최적화된 변환 시간과 의미적 적합성을 보장하지는 못하지만, 그 값에 근사시키기 위한 경험적 방법을 사용한다. 이러한 변환 시간 및 적합성의 보장과 함께 보다 나은 규칙 작성의 용이성을 제공하는 것은, 관리적 측면에서 중요한 이점을 제공할 것이다.

Ontolingua, OntoMorph, 제안된 방법 모두 이전 온톨로지와 새로운 온톨로지 사이의 변환 규칙을 작성한다면 기존 어플리케이션에는 영향을 주지 않게 되므로, 이전 온톨로지와의 상호 운용성을 보장한다.

7. 결 론

본 논문에서는 모바일과 유비퀴터스 컴퓨팅에서 응용되는 RDF 온톨로지 메시지 데이타에 대하여, 의미적 상호 운용성의 문제를 해결하기 위한 하나의 방법으로 실시간 온톨로지 변환 시스템에 대하여 소개하였다. 이러한 온톨로지 변환 시스템의 주요한 문제는 (1) 변환 규칙의 작성, 유지 보수, 관리가 얼마나 용이하고, (2) 변환 시간이 얼마나 신속하며, (3) 또한 변환 후의 의미적 손실이 발생하지 않아야 한다는 것이다. 물론 서로 다른 한 쌍의 온톨로지에 대하여, 관리자가 직접 수작업으로 변환 규칙을 작성하면 의미적 손실이 없는 최적의 변환 규칙을 작성할 수 있지만, 이러한 작업이 다수의 온톨로지에 대하여 이루어 져야 함을 고려하면, 그러한 수작업은 쉽지 않은 일이다. 또한 새로운 온톨로지가 만들어지고 혹은 기존의 온톨로지가 갱신될 경우(온톨로지 버전 관리 문제), 이들 사이의 변환 규칙을 직접 작성하고 관리하는 것은 어려운 작업이 될 것이다. 따라서 본 연구에서는 연쇄 조합 변환 규칙 생성의 개념을 통하여 이러한 직접 변환 규칙 작성의 어려움을 개선해 보고자 하였다. 이의 구현은 XML 데이타 모델과 XQuery 질의의 중첩의 특성을 이용하여 구현될 수 있었다. 몇 가지 실험 결과 및 기존 연구와의 비교 분석은 연쇄 조합 변환 규칙에 의한 온톨로지 변환이 다분히 신속 정확하면서도 규칙 작성의 용이성을 제공함을 보여 준다.

참 고 문 헌

[1] Resource Description Framework (RDF), http://www.w3.org/RDF/

[2] Web Ontology Language (OWL), http://www.w3.org/2004/OWL/

[3] OMA User Agent Profile v1.1, http://www.openmobilealliance.org/release_program/docs/UAProf/OMA-WAP-UAProf-V1_1-20021212-C.pdf

[4] H. Chen, F. Perich, T. Finin, and A. Joshi, "SOUPA: Standard Ontology for Ubiquitous and Pervasive Applications," In Proc. of the First Annual International Conference on Mobile and Ubiquitous Systems: Networking and Service, Boston, MA, Aug. 2004.

[5] H. Chen, T. Finin, A. Joshi, "An ontology for context-aware pervasive computing environments," Knowledge Engineering Review - Special Issue on Ontologies for Distributed Systems, Cambridge University Press, 2004.

[6] D. Dou, D. McDermott, and P. Qi, "Ontology translation by ontology merging and automated reasoning," In Proc. of EKAW Workshop on Ontologies for Multi-Agent Systems, 2002.

[7] H. Chalupsky, "OntoMorph: A Translation System for Symbolic Knowledge," In Proc. of KR 2000, Breckenridge, Colorado, USA, pp. 471-482, 2000.

[8] D. L. McGuinness, R. Fikes, J. Rice, S. Wilder, "An Environment for Merging and Testing Large Ontologies," In Proc. of KR 2000, Breckenridge, Colorado, USA, pp. 483-493, 2000.

[9] F. N. Noy and M. A. Musen, "The PROMPT suite: interactive tools for ontology merging and mapping," International Journal of HumanComputer Studies, 59(6), pp. 983-1024, 2003.

[10] T. Gruber, "Ontolingua: A Translation Approach to Providing Portable Ontology Specifications," Knowledge Acquisition, 5(2), pp. 199-220, 1993.

[11] E. Mena, A. Illarramendi, V. Kashyap, A. Sheth, "OBSERVER: An approach for query processing in global information systems based on interoperation across pre-existing ontologies," International journal on Distributed And Parallel Databases (DAPD), 8(2), pp. 223-271, 2000.

[12] P. Mitra, G. Wiederhold, S. Decker, "A scalable framework for the interoperation of information

sources," In Semantic Web Working Symposium, pp. 317-329, 2001.
[13] A. Doan, J. Madhavan, P. Domingos, A. Halvey, "Learning to map between ontologies on the semantic web," In Proc. of the 11th International Conference on World Wide Web, pp. 662-673, 2002.
[14] Composite Capability/Preference Profiles (CC/PP): Structure and Vocabularies 1.0, http://www.w3.org/TR/2004/REC-CCPP-struct-vocab-20040115/
[15] XQuery 1.0 and XPath 2.0 Functions and Operators, http://www.w3.org/TR/xpath-functions/
[16] XQuery 1.0: An XML Query Language, http://www.w3.org/TR/xquery/
[17] Sample ontologies at Yale DAML project, http://www.cs.yale.edu/homes/dvm/daml/bib-ont.daml
[18] Document-ont v 1.0 at SHOE and DAML, http://www.cs.umd.edu/projects/plus/DAML/onts/docmnt1.0.daml
[19] ATLAS Homework Ontologies, http://www.daml.ri.cmu.edu/ont/homework/cmu-ri-publications-ont.daml, http://www.daml.ri.cmu.edu/ont/homework/atlas-publications.daml
[20] Abacus Relational XQuery, http://216.154.221.184/products/relationalxquery/productfeatures.jsp
[21] JSR-000225 XQuery API for JavaTM (XQJ) (Close of Early Draft Review: 9 July 2004), http://jcp.org/aboutJava/communityprocess/edr/jsr225/
[22] J. Heflin and J. Hendler, "Dynamic ontologies on the web," In Proc. of the Seventeenth National Conference on Artificial Intelligence (AAAI-2000), pp. 443-449, 2000.

김 재 훈
1997년 건국대학교 전자계산학과 공학사 1999년 건국대학교 컴퓨터·정보통신공학과 공학석사. 2005년 서강대학교 컴퓨터학과 공학박사. 2005년 3월~2006년 9월 삼성전자 정보통신총괄 통신연구소 책임연구원. 2006년 9월~현재 서강대학교 컴퓨터학과 BK 21 계약교수. 관심분야는 시맨틱 웹, 웹 데이타베이스, 데이타베이스 보안임

박 석
정보과학회논문지 : 데이타베이스
제 34 권 제 1 호 참조

부록 1. T2.xq: mobile2 -〉 mobile1의 직접 변환 규칙

```
declare namespace rdf = "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
declare namespace prf1 = "mobile1_schema1#"
declare namespace prf2 = "mobile2_schema1#"

declare function local:mk_mobile1_BluetoothProfile($btpro as element(prf2:SupportedFunction))
as element(rdf:Bag)
{
   <rdf:Bag>                                          <!-- code  function_name mapping -->
   {                                                  <!-- 3090  'headset' -->
      for  $a in $btpro/rdf:Bag/rdf:li,               <!-- 3355  'lanaccess' -->
           $b in collection('MAPPING.T1')/T1          <!-- T1 is a mapping table -->
      where $a/text() = $b/C1/text()
      return
         <rdf:li>{$b/C2/text()}</rdf:li>
   }
   </rdf:Bag>
};

declare function log($num1 as xs:integer, $num2 as xs:integer) as xs:integer external;

<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#" xmlns:prf1="mobile1_schema1#">
   <rdf:Description rdf:ID="Device1Profile">
      <prf1:component>
         <prf1:HardwarePlatform>
            <prf1:BluetoothProfile>
            {
               let $a := doc("source.rdf")
               return    local:mk_mobile1_BluetoothProfile($a//prf2:SupportedFunction)
            }
            </prf1:BluetoothProfile>
            <prf1:ScreenSize>
            {
               let $a := doc("source.rdf")
               return
                  concat(concat($a//prf2:Screen/prf2:Width/text(), "*"), $a//prf2:Screen/prf2:Height/text())
            }
            </prf1:ScreenSize>
```

```
            <prf1:Model>R999</prf1:Model>
            <prf1:BitsPerPixel>
            {
               let $a := doc("source.rdf")
               return log(2, $a//prf2:Screen/prf2:Color/text())
            }
            </prf1:BitsPerPixel>
            <prf1:ColorCapable>
            {
               let $a := doc("source.rdf")
               return
                  $a//prf2:Screen/prf2:ColorCapable/text()
            }
            </prf1:ColorCapable>
            <prf1:ImageCapable>
            {
               let $a := doc("source.rdf")//prf2:CcppAccept
               return
                  if(some $b in $a//rdf:li satisfies contains($b/text(), "gif"))
                  then "YES" else "NO"
            }
            </prf1:ImageCapable>
                  : :          : :          : :
         </prf1:HardwarePlatform>
      </prf1:component>
      <prf1:component>
         <prf1:SoftwarePlatform>
            <prf1:CcppAccept>
            {
               let $a := doc("source.rdf")//prf2:CcppAccept
               return $a/rdf:Bag
            }
            </prf1:CcppAccept>
                  : :          : :          : :
         </prf1:SoftwarePlatform>
                  : :          : :          : :
      </prf1:component>
   </rdf:Description>
</rdf:RDF>
```

부록 2. T3.xq: mobile3 -〉 mobile2의 직접 변환 규칙

```
declare namespace rdf = "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
declare namespace prf2 = "mobile2_schema1#"
declare namespace prf3 = "mobile3_schema1#"

declare function local:mk_mobile2_SupportedFunction($bthead as xs:string, $btlan as xs:string)
as element(rdf:Bag)
{
   <rdf:Bag>
   {
      let $a := $bthead
      return
         if( not(compare($a, "YES")))
         then  <rdf:li>3090</rdf:li> else ()
   }
   {
      let $a := $btlan
      return
         if( not(compare($a, "YES")))
         then  <rdf:li>3355</rdf:li> else ()
   }
   </rdf:Bag>
};

declare function power($num1 as xs:integer, $num2 as xs:integer) as xs:integer external;

<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#" xmlns:prf2="mobile2_schema1#">
   <rdf:Description rdf:ID="Device2Profile">
      <prf2:component>
         <prf2:HW>
            <prf2:BT>
               <prf2:ModelInfo>BT2006</prf2:ModelInfo>
               <prf2:SupportedFunction>
               {
```

```
                let $a := doc("source.rdf")
                return local:mk_mobile2_SupportedFunction($a//prf3:BluetoothHeadset/text(),
                                                          $a//prf3:Bluetoothlanaccess/text())
            }
            </prf2:SupportedFunction>
        </prf2:BT>
        <prf2:Screen>
            <prf2:Width>
            {
                let $a := doc("source.rdf")
                return substring-before($a//prf3:LCDSize/text(), "*")
            }
            </prf2:Width>
            <prf2:Height>
            {
                let $a := doc("source.rdf")
                return substring-after($a//prf3:LCDSize/text(), "*")
            }
            </prf2:Height>
            <prf2:Color>
            {
                let $a := doc("source.rdf")
                return power(2, $a//prf3:ColorPallete/text())
            }
            </prf2:Color>
            <prf2:ColorCapable>
            {
                let $a := doc("source.rdf")
                return $a//prf2:ColorSupport/text()
            }
            </prf2:ColorCapable>
        </prf2:Screen>
            : :          : :          : :
    </prf2:HW>
</prf2:component>
<prf2:component>
    <prf2:SW>
        <prf2:CcppAccept>
        {
            let $a := doc("source.rdf")//prf3:CcppAccept
            return <rdf:Bag>{
                    for $b in $a/rdf:Bag/rdf:li
                    where not(contains($b/text(), "mobile3"))
                    return <rdf:li>{$b/text()}</rdf:li>}</rdf:Bag>
        }
        </prf2:CcppAccept>
            : :          : :          : :
    </prf2:SW>
</prf2:component>
            : :          : :          : :
</rdf:Description>
</rdf:RDF>
```

부록 3. T4.xq: mobile3 -〉 mobile1의 연쇄 조합 변환 규칙

```
declare namespace rdf = "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
declare namespace prf1 = "mobile1_schema1#"
declare namespace prf2 = "mobile2_schema1#"
declare namespace prf3 = "mobile3_schema1#"

declare function local:mk_mobile1_BluetoothProfile($btpro as element(prf2:SupportedFunction))
as element(rdf:Bag)
{
    <rdf:Bag>                                          <!-- code  function_name mapping -->
    {                                                  <!-- 3090  'headset' -->
        for  $a in $btpro/rdf:Bag/rdf:li,              <!-- 3355  'lanaccess' -->
             $b in collection('MAPPING.T1')/T1         <!-- T1 is a mapping table -->
        where $a/text() = $b/C1/text()
        return
            <rdf:li>{$b/C2/text()}</rdf:li>
    }
    </rdf:Bag>
};
```

```
declare function local:mk_mobile2_SupportedFunction($bthead as xs:string, $btlan as xs:string)
as element(rdf:Bag)
{
   <rdf:Bag>
   {
      let $a := $bthead
      return
         if( not(compare($a, "YES")))
         then  <rdf:li>3090</rdf:li> else ()
   }
   {
      let $a := $btlan
      return
         if( not(compare($a, "YES")))
         then  <rdf:li>3355</rdf:li> else ()
   }
   </rdf:Bag>
};

declare function power($num1 as xs:integer, $num2 as xs:integer) as xs:integer external;
declare function log($num1 as xs:integer, $num2 as xs:integer) as xs:integer external;

<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#" xmlns:prf1="mobile1_schema1#">
   <rdf:Description rdf:ID="Device1Profile">
      <prf1:component>
         <prf1:HardwarePlatform>
            <prf1:BluetoothProfile>
            {
               let $a := doc("source.rdf")
               return    local:mk_mobile1_BluetoothProfile(
                              local:mk_mobile2_SupportedFunction($a//prf3:BluetoothHeadset/text(),
                                                   $a//prf3:Bluetoothlanaccess/text()))
            }
            </prf1:BluetoothProfile>
            <prf1:ScreenSize>
            {
               let $a := doc("source.rdf")
               return
                 concat(concat(substring-before($a//prf3:LCDSize/text(), "*"), "*"),
                              substring-after($a//prf3:LCDSize/text(), "*"))
            }
            </prf1:ScreenSize>
            <prf1:Model>R999</prf1:Model>
            <prf1:BitsPerPixel>
            {
               let $a := doc("source.rdf")
               return log(2, power(2, $a//prf3:ColorPallete/text()))
            }
            </prf1:BitsPerPixel>
            <prf1:ColorCapable>
            {
               let $a := doc("source.rdf")
               return
                  $a//prf2:ColorSupport/text()
            }
            </prf1:ColorCapable>
            <prf1:ImageCapable>
            {
                let $a := (
                   let $c := doc("source.rdf")//prf3:CcppAccept
                   return <rdf:Bag>{
                      for $d in $c/rdf:Bag/rdf:li
                      where not(contains($d/text(), "mobile3"))
                      return <rdf:li>{$d/text()}</rdf:li>}</rdf:Bag>
                )
                return
                   if(some $b in $a//rdf:li satisfies contains($b/text(), "gif"))
                   then "YES" else "NO"
            }
            </prf1:ImageCapable>
               : :          : :          : :
         </prf1:HardwarePlatform>
```

```
    </prf1:component>
    <prf1:component>
      <prf1:SoftwarePlatform>
        <prf1:CcppAccept>
        {
          let $a := (
            let $b := doc("source.rdf")//prf3:CcppAccept
            return <rdf:Bag>{
              for $c in $b/rdf:Bag/rdf:li
              where not(contains($c/text(), "mobile3"))
              return <rdf:li>{$c/text()}</rdf:li>}</rdf:Bag>
          )
          return $a/rdf:Bag
        }
        </prf1:CcppAccept>
            : :          : :          : :
      </prf1:SoftwarePlatform>
            : :          : :          : :
    </prf1:component>
  </rdf:Description>
</rdf:RDF>
```

부록 4. XQuery 문법의 EBNF 표기의 일부분

```
[31] Expr ::= ExprSingle ("," ExprSingle)*
[32] ExprSingle ::= FLWORExpr
                  | QuantifiedExpr
                  | TypeswitchExpr
                  | IfExpr
                  | OrExpr
[33] FLWORExpr ::=
        (ForClause | LetClause)+ WhereClause? OrderByClause? "return" ExprSingle
[34] ForClause ::=
        "for" "$" VarName TypeDeclaration? PositionalVar? "in" ExprSingle ("," "$"
        VarName TypeDeclaration? PositionalVar? "in" ExprSingle)*
[35] PositionalVar ::= "at" "$" VarName
[36] LetClause ::=
        "let" "$" VarName TypeDeclaration? ":=" ExprSingle ("," "$" VarName
        TypeDeclaration? ":=" ExprSingle)*
[37] WhereClause ::= "where" ExprSingle
[38] OrderByClause ::= (("order" "by") | ("stable" "order" "by")) OrderSpecList
[39] OrderSpecList ::= OrderSpec ("," OrderSpec)*
[40] OrderSpec ::= ExprSingle OrderModifier
[41] OrderModifier ::=
        ("ascending" | "descending")? ("empty" ("greatest" | "least"))? ("collation"
        URILiteral)?
[93] FunctionCall ::= QName "(" (ExprSingle ("," ExprSingle)*)? ")"
```