RDFS를 이용한 관계표현 확장

정관호*, 공현장*, 유해도*, 황명권*, 김판구**
* 조선대학교 전자계산학과
**조선대학교 컴퓨터공학부
khjung@mina.chosun.ac.kr

Extension of Relationship in RDF Schema

Kwan-Ho Jung*, Hyun-Jang Kong*, Hai-Tao Liu*,
Hwang-Myung Gwun*, Pan-Koo Kim**
*Dept of Computer Science, Chosun University
**Dept of Computer & Information Engineering, Chosun University

요 약

온톨로지의 구축에서는 각 개념간의 관계의 정의가 매우 중요하며, 이를 표현하고 정의하는 많은 일련의 과정이 진행되고 있다. RDF 스키마내부에 시간관계 개념인 “before”를 정의하고 에이전트가 ”before” 시간관계를 해석 할 수 있도록 Logic정의를 시도하였다. 또한 시간개념을 사용하여 온톨로지를 구출할 때 발생할 수 있는 이점에 대해 살펴보았다.

1. 서론

최근 월드 와이드 웹 컨소시움 (W3C; World Wide Web Consortium)은 RDF와 웹 온톨로지 언어인 OWL을 공식으로 인정하였다고 발표하였다.[1] W3C의 이번 권고안은 지금까지 연구개발 프로젝트에서 진행되어왔던 시맨틱 웹의 기술이 상용 레벨의 플랫폼으로 그 모습을 세상에 드러내는 것이라 할 수 있으며 엔터프라이즈 통합, 의료 의사 결정지원과 같은 분야에서 새로운 제품이 급성장할 가능성을 보여 주는 것이었다. 시맨틱 웹의 핵심은 온톨로지며 또한 온톨로지의 핵심은 자원과 자원간의 관계정의다. 관계정의에 따라 자원의 특성과 성격이 달라질 수 있다. W3C에서는 온톨로지 구축 시 관계를 정의 할 수 있는 관계들을 지속적으로 개발해 나가고 있지만 현재까지는 시간관계를 표현할 수 있는 방안은 존재하지 않고 있다. 일상생활에서는 물론 전문영역으로 세분화 되어 질수록 시간적인 표현은 더욱 요구되어지고 있다. 간단한 예로, 역사는 시간이라는 큰 맥락 하에 사건들이 발생한다. 그러므로 역사의 한 중요 부분을 온톨로지로 구축할 시, 시간적 순서를 바탕으로 사건들을 연결해야 하는 건 당연한 일이다. 하지만 아직까지는 시간적인 개념을 표현 할 수 있는 시간관계 표현이 없다. 그래서 본 논문에서는 시간적 개념을 지닌 자원들간의 관계에 대해 표현하고자 하였다.

2. RDF에서의 관계 및 제약

2.1. 시맨틱 웹 레이어

시맨틱 웹의 레이어는 URI(Universal Resource Identifiers)와 Unicode를 기반으로 상위에 XML, RDF, Ontology, Logic, Proof, Trust로 나누워져 있다[2]. XML은 문서요소들 사이에서의 구조적 정의를 나타내며 RDF는 정보자원이나 자원의 타입을 기술하는 언어로 의미표현을 위한 수단으로 사용되어진다. RDF는 컴퓨터가 이해할 수 있는 형태의 정보를 응용프로그램 사이에서 교환하기 위한 수단으로 사용되어진다. Ontology는 특정 도메인에 대한 공유되는 일반적인 이해, 개념의 표현, 개념과 관계에 대한 공식적인 기술을 의미한다. Logic계층은 메타데이터의 논리기반을 제공하고 또한 S/W 및 에이전트는 제공되는 논리기반을 바탕으로 의미해석과

추론한다. Proof는 Logic 단계의 추론 및 관계도출에 대한 증명을 할 수 있는 단계를 말하며, 이 단계를 걸쳐 도출된 결과가 믿을 수 있는, 신뢰할 수 있는 결과임을 명시하기위해 Trust를 최상위 단계에 두고 있다.

2.2. 자원간의 관계 및 제약을 위한 RDF스키마

RDF계층을 세분화 해보면 RDF Syntax 부분과 RDF Schema 부분으로 나눌 수 있다. RDF 스키마는 자원의 특성을 기술하기 위해 사용될 수 있는 자원의 집합이며 또한 자원을 기술하기 위해 사용되는 어휘들을 정의하고 자원과 자원 사이의 관계와 제약조건을 기술하기 위한 기본적인 특성을 기술하고 정의하고 있다[3][5].

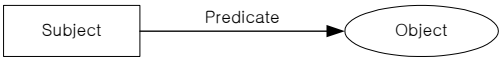
[표 1] RDF 스키마 내부에서 “subPropertyOf” 정의

```
<rdf:Property rdf:about="http://www.w3.org/2000/01/rdf-schema#subPropertyOf">
  <rdfs:isDefinedBy rdf:resource="http://www.w3.org/2000/01/rdf-schema#"/>
  <rdfs:label>subPropertyOf</rdfs:label>
  <rdfs:comment>The subject is a subproperty of a property.</rdfs:comment>
  <rdfs:range
        rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
  <rdfs:domain
        rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
</rdf:Property>
```

RDF 스키마에 존재하는 제약 및 조건들은 확장이 가능하다. 하지만 확장을 위해선 우선 그것들에 대한 Logic적 증명이 뒤받침 되어야 한다. 현재 구축되어있는 조건 및 제약들은 Description Logic 및 First Oder Logic, F-Logic등을 사용하여 증명을 하고 있다[4].

3. 시간관계를 위한 RDF 스키마 정의

RDF 구문은 Subject, Predicate, Object로 구성되어있다. Predicate은 Property로 불리기도 하며 자원간의 관계를 나타낸다[4][6].

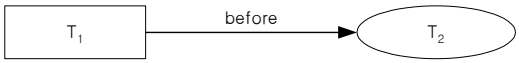


[그림 1] RDF Graph Data Model

Predicate은 URI(Uniform Resource Identifier)가 될 수 있고 "subPropertyOf"와 같이 자원간의 관계를 나타낼 수 도 있다. 시간관계를 나타내기 위해 시간적 개념을 내포하고 있는 T_1과 T_2 사이에 Predicate으로 관계를 정의해 주어야한다. 본 논문에서는 많은 시간관계 중에서 시간의 전후관계를 표현하는데 주안점을 두기로 한다. 시간개념을 내포하고 있는 T_1과 T_2가 존재한다고 가정 했을 경우 이 두 자원을 연결하는 관계를 “before"이라고 명시하면 다음과 같은 조건이 발생한다.

조건 1 : ”before"은 T_1과 T_2가 존재할 때 T_1과 T_2 사이의 전후관계를 나타낸다.

조건 2 : “before"으로 연결된 T_1은 T_2보다 시간적으로 과거의 개념이다.



[그림 2] “before"를 이용한 자원들간의 관계

또한 시간적 개념을 내포하는 T_1과 T_2의 range 와 domain은 class에 속해야 한다. 위의 정의를 바탕으로 RDF 스키마에 관계를 정의하면 다음과 같다.

[표 2] RDF 스키마에 “before"관계 정의

```
<rdf:Property rdf:about="http://www.w3.org/2000/01/rdf-schema#before">
  <rdfs:isDefinedBy rdf:resource="http://www.w3.org/2000/01/rdf-schema#"/>
  <rdfs:label>before</rdfs:label>
  <rdfs:comment>The subject is before than object.</rdfs:comment>
  <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
  <rdfs:domain rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
</rdf:Property>
```

그림[2]를 “before"관계를 이용해 RDF 구문으로 간략하게 표현하면 표[3] 같이 나타낼 수 있다.

[표 3] 시간관계를 이용한 T_1과 T_2의 RDF구문

```
<rdf:RDF xml:lang="en"
 xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
 xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">

 <rdf:Description ID="T1">
  <rdf:type resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
  <rdfs:before rdf:resource="#T2"/>
 </rdf:Description>

 <rdf:Description ID="T2">
  <rdf:type resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
  <rdfs:before rdf:resource="#T1"/>
 </rdf:Description>

</rdf:RDF>
```

4. 시간관계 "before"의 의미적 해석

시간개념을 가진 T_1과 T_2가 있다고 할 때, T_1과 T_2를 시간적 관계로 연결하기 위해선 아래와 같은 조건들이 필요하다.(본 논문에선 많은 시간적 관계

중 전후 관계만으로 한정한다.)
조건 4. T_1과 T_2는 시간적으로 중복되지 않아야 한다.
조건 5. T_1과 T_2는 동일하지 않아야 한다.
조건 6. T_1과 T_2의 Type은 Class다.

위의 조건에 대해서 "before"을 정의하면 다음과 같다. 단 다음과 같은 가정 하에 정의를 한다. (T_1은 T_2보다 시간적으로 과거다)

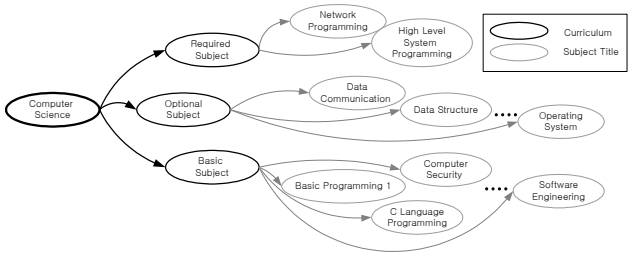
정의 1 : 시간적 개념 T_1과 T_2에 대해
before = $\{T_1,T_2 \mid T_1,T_2 \in \text{Class and } T_1 \cap T_2 = \varnothing \text{ and } T_1 \neq T_2\}$

위의 제약으로는 시간관계(전후관계)를 만족시키지 못 한다. 시간은 어떤 수치적인 크기로 전후를 판단하기는 힘들다. 예를 들어, 과거와 현대라는 단어의 의미만을 놓고 볼 때 과거라는 것을 수치적으로 표현하기 불가능하다. 그 단어가 내포하는 의미를 만족하면서 또한 전후의 관계를 나타내기 위해선 위에서 명시한 제약에 다른 제약이 더 참가되어야 한다. 조건 6 에서 시간개념을 가진 T_1과 T_2의 Type은 클래스여야 한다. T_1과 T_2가 시간적인 개념을 지니고 또한 Type이 Class이므로 클래스 속성을 RDF의 리소스로부터 상속 받을 수 있다. 최종적으로 T_1과 T_2에 대해 Class적 개념으로 전후관계를 도출하면 다음과 같다.

정의 2 : 시간적 개념 T_1과 T_2에 대해
before = $\{T_1,T_2 \mid T_1,T_2 \in \text{Class and } T_1 \cap T_2 = \varnothing \text{ and } T_1 \neq T_2 \text{ and } T_1 \supseteq T_2\}$

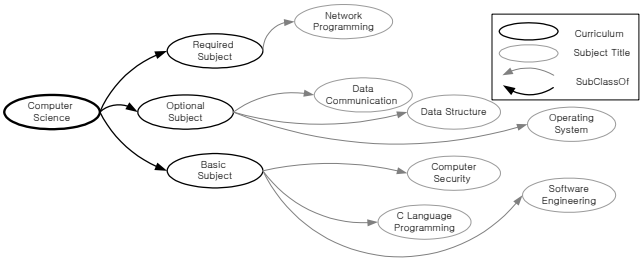
5. 시간관계를 적용한 학과 커리큘럼

이 절에서는 앞 절에서 정의한 시간관계를 이용하여 학과 커리큘럼 온톨로지를 구축해보고자 하였다. 전자계산학과의 교과목을 예를 들어 보도록 하자. 그림[3]은 전자계산학과의 교과목을 간략하게 나타내고 있다.



[그림 3] 일반적인 전자계산학과 Curriculum

전자계산학과에는 기본과목, 선택과목 그리고 필수과목이 있고 각 교과목 밑에는 세부과목들이 존재한다. 만일 어떤 학생이 Network Programming 과목을 듣고자 했을 경우 우선 선택과목에서 Data Communication, Data Structure를 이수해야 하고 또한 Basic Subject과목 중에서 C Language Programming, Software Engineering, Computer Security과목을 이수해야 한다.



[그림 4] Network Programming 과목을 신청하기 위한 하위레벨 구성

그림[4]는 "Network Programming" 과목을 수강하기 위한 선수과목을 나타내고 있다. 각 과목은 교과목이라는 커리큘럼에 속하게 되며 각 커리큘럼은 학과의 정규교과목에 속하게 된다. 일반적인 - 현존하는 관계만을 이용하여 온톨로지를 구축하는 - 방법으로 학과의 커리큘럼을 구축할 때 발생 할 수 있는 문제점들에 대해 살펴보면 다음과 같다.

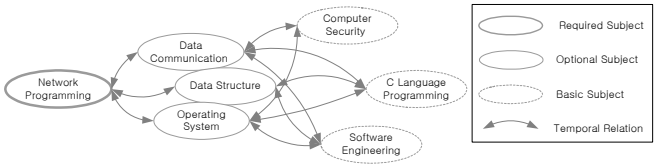
1. 상위의 단계를 신청하기 위해서 어떤 하위과목을 들어야 하는지 구분하기 힘들다. 예를 들어, "Network Programming" 과목을 신청하기 위해서 학생들은 사전에 "Data Communication", "Data Structure", "Operation System" 과목을 사전에 수강해야 하지만 "subClassOf" 관계로 커리큘럼 온톨로지를 구축한다면 어떤 과목을 먼저 수강해야 하는지 알 수 없는 경우가 발생한다.[그림 3] 참조

2. 기본교과목을 수강한 학생이 다음 단계로 어떤 과목을 신청할 수 있는지 쉽게 알기 어렵다. 예를 들어, 한 학생이 "C Language Programming" 과목을 수강한 후 다음 단계로 어떤 과목을 선택해야 하는지 알기 힘들다.[그림 3] 참조

3. 모든 커리큘럼의 과정을 빠르게 알기 힘들다. 예를 들면, 그림[3]에서 "High Level System

Programming" 과목을 수강하기 위해선 Basic Subject 과정과 Optional Subject 과정에서 어떤 과들을 수강해야 하는지 알기 힘들다.

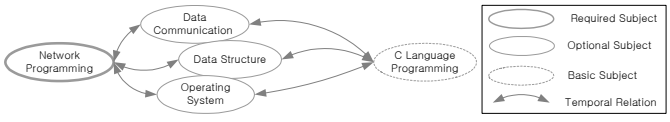
위에 나열했던 문제점들을 해결하기 위해선 각 과목을 "subClassOf" 관계가 아닌 시간적 개념을 가진 관계로 묶어 두어야 한다.



[그림 5] 시간적 관계를 이용한 과목들의 연결

그림[3]은 각각의 과목을 시간적 개념을 지닌 관계로 연결하여 표현해 놓았다. "Network Programming" 과목을 수강하기 위해선 Basic Subject 과정과 Optional Subject 과정에서 각각 어떤 과목들을 들어야 하는지 쉽게 알 수 있다. 시간적 관계로 각 과목을 연결 했을 경우 얻을 수 있는 장점들을 살펴보면 다음과 같다.

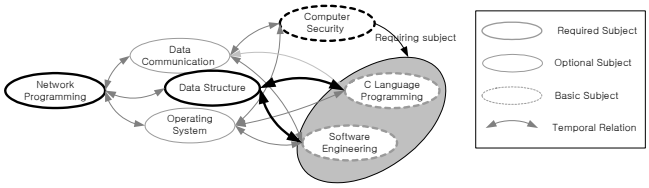
1. 상위 단계의 과목을 수강하기 위해선 하위에 단계에서 어떤 과목을 수강해야 하는지 쉽게 알 수 있다. 예를 들어, 그림[4]에서 "Network Programming" 과목을 수강하기 위해선 Basic Subject 과정에서 "Data Communication", "Data Structure" and "Operation System" 과목을 수강해야 한다는 걸 쉽게 알 수 있다. 또한 Basic Subject 과정에선 "C Language Programming" 과목만 들어도 된다는 걸 쉽게 알 수 있다.



[그림 6] 시간적 관계로 표현한 Network Programming 과목 코스

2. Basic Subject 단계 다음에 어떤 과목들을 수강신청해야 하는지 쉽게 알 수 있다. 예를 들어, "C Language Programming" 과목을 수강한 학생은 다음 단계로 "Data Communication", "Data Structure" and "Operating System" 과목을 수강 신청 할 수 있다는 걸 쉽게 알 수 있다.

3. 전 과정의 커리큘럼을 빠르게 알 수 있다. 최상위 단계에서 수강하고자 하는 과목을 중심으로 하여 하위 과목들이 나열되어 있고 또한 각 과정별로 어떤 과목을 수강 해야지 상위로 과목을 수강 할 수 있는지 빠르게 알 수 있다. 이는 각 과목들이 전후 관계로 되어 있기 때문이다. 또한 만약 Basic Subject 단계에서 "C Language Programming" 만 수강한 학생이 "Data Structure"과목을 듣고자 할 때 반드시 "C Language Programming" 또는 "Software Engineering" 들어야 한다는 것도 쉽게 알 수 있다.



[그림 7] Data Structure 과목을 수강하기 위한 하부 필요과목

6. 결론 및 향후연구

본 논문에서는 RDF 스키마 내부에 시간관계를 표현 할 수 있는 “before"이라는 관계를 정의하였고 의미적인 해석까지 해보았다. 또한 시간적 개념을 가진 자원들간에 전후관계를 명시하는데 주력했다. 하지만 시간관계는 전후관계 뿐만 아니라 시간의 중복, 시간의 인접 등 다양한 관계들이 존재하고 있다. 이런 다양한 시간관계의 표현과 증명이 향후연구로 남아있다.

참고문헌

[1]http://www.w3.org/2004/02/voicexml2-pressrelease.html.en
[2]http://www.w3.org/2004/Talks/0120-semweb-umich/Overview.html
[3]http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/
[4]http://www.w3.org/TR/2004/REC-rdf-schema-20040210/
[5]http://www.w3.org/TR/2004/REC-rdf-syntax-grammar-20040210/
[6]http://www.w3.org/TR/2004/REC-rdf-primer-20040210/