韓國IT서비스學會誌
第7卷 第3號
2008年 9月

개념격자를 이용한 온톨로지 오류검출기법

황 석 형*

An Approach for Error Detection in Ontologies Using Concept Lattices

Suk-Hyung Hwang*

■ Abstract ■

The core of the semantic web is ontology, which supports interoperability among semantic web applications and enables developer to reuse and share domain knowledge. It used a variety of fields such as Information Retrieval, E-commerce, Software Engineering, Artificial Intelligence and Bio-informatics. However, the reality is that various errors might be included in conceptual hierarchy when developing ontologies. Therefore, methodologies and supporting tools are essential to help the developer construct suitable ontologies for the given purposes and to detect and analyze errors in order to verify the inconsistency in the ontologies.

In this paper we propose a new approach for ontology error detection based on the Concept Lattices of Formal Concept Analysis. By using the tool that we developed in this research, we can extract core elements from the source code of Ontology and then detect some structural errors based on the concept lattices. The results of this research can be helpful for ontology engineers to support error detection and construction of "well-defined" and "good" ontologies.

Keyword : Ontology, Concept Lattice, Error Detection, Formal Concept Analysis

논문투고일 : 2008년 04월 18일 논문수정완료일 : 2008년 09월 03일 논문게재확정일 : 2008년 09월 18일
* 선문대학교 컴퓨터공학부

1. 서 론

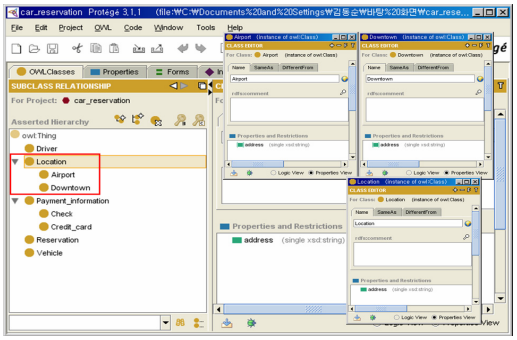
온톨로지는 관심영역에 관한 공유지식의 표현으로서, 시멘틱 웹의 상호운용성을 지원하기 위한 중요한 핵심기술로서 중요한 역할을 수행하고 있다[1, 4]. 또한, 온톨로지는 의료정보, 바이오정보, 인공지능, 에이전트, 유비쿼터스 컴퓨팅, 전자상거래 등의 다양한 IT분야에서 대규모적인 지식 공유와 재사용을 위해 응용되고 있다.

온톨로지를 기술할 수 있는 다양한 언어들(RDF, OWL, DAML+OIL 등)[6]이 제안되어있다. RDF (Resource Description Framework)[10]는 W3C에서 발표한 메타데이터의 기술과 교환을 위한 표준이며, 다양한 메타데이터 사이의 상호운용을 위해서 의미와 구조, 그리고 구문에 대한 공통적인 규칙을 제공한다. 또한, OWL(Web Ontology Language)[8]은 RDF와 RDF Schema의 한계를 보완하고자 개발된 언어로서, 풍부한 어휘와 형식적 의미론을 포함하고 있기 때문에 기계해석이 가능한 웹 콘텐츠를 저작하는 경우, XML이나 RDF보다 우수하여, 현재 온톨로지 언어의 표준으로 자리잡고 있다. 그러나 온톨로지 표현 언어를 사용하여 사람이 직접 온톨로지를 작성하기에는 너무나 복잡하고, 어렵기 때문에 쉽게 온톨로지를 구축하고 편집할 수 있는 도구들이 활용되고 있다.

현재, 다양한 온톨로지 구축도구들이 개발되어 있으며, 특히, Protégé[9]는 쉬운 사용방법과 간편한 GUI를 제공함으로서 온톨로지를 쉽게 구축하고, 편집할 수 있으며, OWL, RDF, XML 등의 다양한 온톨로지 언어와 포맷을 지원하고 있을 뿐만 아니라 확장 가능한 API를 제공함으로서 다양한 플러그인 기능들이 추가되어 있다. 한편, OntoEdit [12]는 온톨로지를 구축하고, 편집하는 기본적인 기능과 더불어 데이터베이스 시스템과 연동할 수 있는 기능을 제공하며, RDF와 DAML+OIL을 지원한다. 또한, WebODE[7]는 서버와 클라이언트를 통해서 여러 사람들이 협동적으로 온톨로지를 구축할 수 있는 환경을 제공한다.

그러나 대부분의 온톨로지들은 도메인전문가나 온톨로지 개발자들이 Protégé와 같은 도구를 사용하여 전문가들의 의견과 지식, 경험 등을 종합하여 수작업으로 구축되어 지고 있다. 그럼에도 불구하고, 구축하려는 온톨로지의 도메인으로부터 온톨로지의 구성요소를 추출하기 매우 어려우며, 이러한 온톨로지의 구성요소를 가지고 온톨로지를 모델링 하는 작업 또한 쉽지 않다[7]. 따라서, 좋은 온톨로지 구축도구를 사용한다고 할지라도 실용적이고, 도메인의 정보를 정확하게 반영한 온톨로지를 구축하는 것은 쉽지 않다.

특히, 최근의 연구결과[2, 11]에 따르면, 구축하려는 온톨로지의 도메인으로부터 온톨로지의 구성요소를 추출하기 매우 어려우며, 이러한 온톨로지의 구성요소를 가지고 온톨로지를 모델링 하는 작업 또한 쉽지 않다. 특히, 대다수의 온톨로지 개발관련 프로젝트에서 온톨로지 개발자들이 개발방법론을 적용하지 않고, 자신의 경험에만 의존하여 온톨로지를 수작업으로 구축하고 있는 것으로 보고되고 있다. 따라서 수작업에 의해 구축된 온톨로지에는 여러 가지 오류들이 포함되어 있을 가능성이 매우 높을 수 있다.



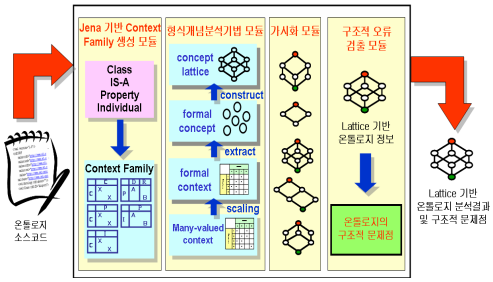
[그림 1] Protégé에서 작성된 Car Reservation 온톨로지

예를 들어, [그림 1]은 Protégé의 확장기능으로 구현된 Prompt(http://protege.stanford.edu/plugins/prompt/prompt.html)에서 예제로 제공되고 있는 Car Reservation 온톨로지를 나타내고 있다. Car

Reservation 온톨로지에서는 Location 클래스의 하위클래스로 Airport와 Downtown 클래스들이 선언되어 있다. 그러나 Location은 1개의 프로퍼티(address)가 정의되어 있으며, Airport와 Downtown에 상속되고 있다. 이러한 경우, Location의 하위클래스로 선언된 Airport와 Downtown은 Location과 비교하여 다른 프로퍼티가 추가정의 되어있지 않기 때문에, 프로퍼티중심적인 디자인관점에서 엄밀한 포섭관계가 성립하지 않는다. 따라서 Location과 Airport, 그리고 Location과 Downtown사이에 엄밀한 포섭관계가 성립하려면 Airport와 Downtown 각각에 새로운 프로퍼티들이 추가되어야 한다. 수작업에 의한 온톨로지 구축에 있어서 위와 같은 문제는 빈번하게 발생할 수 있으므로, 구축된 온톨로지 소스에 대하여 이와 같은 문제점을 찾아내기 위한 분석이 필요하다.

본 논문에서는, 이와 같은 문제점을 해결하기 위하여, 형식개념분석기법(Formal Concept Analysis) [3, 5]에서 제공하는 개념계층구조를 토대로 하는 온톨로지 오류검출기법을 제안한다. 형식개념분석기법은 개념격자(Concept Lattice)라는 수학적 개념계층구조화 모델을 기반으로 하는 데이터분석기법으로서, 개념적인 데이터분석에 의한 지식추출 및 포섭관계를 나타내는 개념계층의 구조화가 가능하다. 형식개념분석기법의 개념격자를 기반으로 하여 OWL의 기본구성요소들을 표현, 분석하기위한 모델을 정의하고, OWL로 작성된 온톨로지 소스코드로부터 온톨롤지의 기본요소들을 분석하여 온톨로지에 포함된 오류를 자동으로 파악하고 수정 보완할 수 있는 방법을 제안하였다. 또한, 본 연구에서 제안된 온톨로지 오류검출기법을 토대로, 입력된 온톨로지 소스코드에 대하여 형식개념분석을 수행하여 온톨로지에 내재된 에러를 자동으로 검출하는 지원도구[그림 2]를 개발하고, 몇가지 대표적인 온톨로지들을 분석하고 오류를 검출하는 사례를 소개한다.

본 논문은 다음과 같이 구성되어 있다. 제 2장에서는 온톨로지의 기본적인 구성요소들과 형식개념



[그림 2] 본 연구에서 개발된 온톨로지 분석 도구의 구조

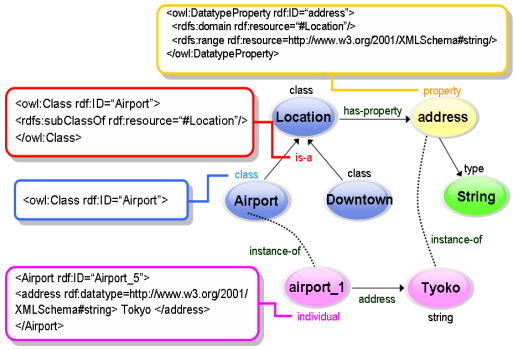
분석기법을 소개한다. 제 3장에서는 본 연구에서 개발된 온톨로지 오류검출에 대해서 기술하고, 제 4장에서는 오류검출에 관한 실험사례를 설명한다. 그리고 제 5장에서는 본 논문의 의의와 결론을 논의한다.

2. 온톨로지의 기본구성요소와 형식개념분석기법

2.1 온톨로지의 기본구성요소

온톨로지는 공유된 개념화에 대한 정형화되고 명시적으로 기술해놓은 지식모델을 말한다. 즉, 특정 도메인의 개념들을 정의하고, 그들 사이의 관계를 계층적으로 표현하고 있으며, 추가적으로 추론 규칙이 포함되어 있다[1, 4]. 온톨로지를 표현하기 위해 스키마와 구문구조 등을 정의한 언어가 온톨로지 언어이며, XML 기반의 XOL(Ontology Exchange Language), OML(Ontology Markup Language), SHOE(Simple HTML Ontology Extensions)와 W3C에서 제정한 RDF와 OIL, DAML), DAML+OIL, OWL 등이 있다[6].

온톨로지들은 다양한 언어로 표현되고 있음에도 불구하고 구조적으로 많은 유사성을 보이고 있다. 대부분의 온톨로지들은 Individuals(기본적인 또는 기저적인 개념들), Classes(개념들/객체들의 집합, 컬랙션 또는 타입), Attribute(개념/객체가 보유하거나 공유할 수 있는 속성 또는 특성), 그리고 Re-



[그림 3] Car Reservation 온톨로지에 대한 OWL소스코드

lations(개념/객체가 다른 개념/객체와 연관되는 방법)을 기술하고 있다. 특히, Protégé와같은 도구에 의해 작성된 온톨로지의 각 구성요소들은 OWL언어를 기반으로, owl : Class키워드를 사용하여 클래스들을 정의하고, 그들 사이의 is-a 관계를 sub ClassOf를 이용하여 정의한다. OWL에는 2종류의 property(datatype property, object property)가 제공되며, datatype property는 문자열이나 숫자타입의 range속성을, 한편으로 object property는 individual을 range값으로 갖는 프로퍼티를 나타낸다. 또한, individual은 특정 클래스에 속하는 인스턴스로서 해당 클래스가 갖는 속성들의 실제 데이터 값을 포함하고 있다. [그림 3]은 Car Reservation 온톨로지에 대한 OWL소스코드로부터 파악할 수 있는 OWL온톨로지의 기본구성요소들의 예를 보여주고 있다.

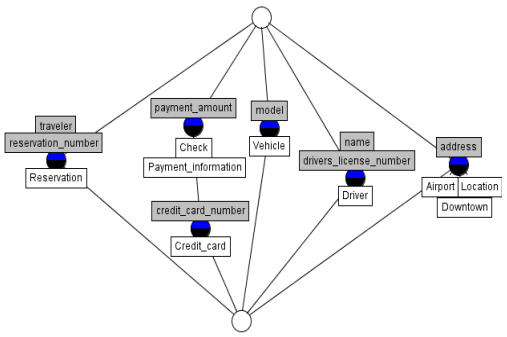
2.2 형식개념분석기법

형식개념분석기법[3, 5]는 개념격자라는 수학적 개념계층모델을 기반으로 하는 데이터클러스터링 기법이다. 형식개념분석의 기본이 되는 구조 Formal context K = (G, M, I)는 객체들(Objects)의 집합 G와 속성들(Attributes)의 집합 M, 그리고 G와 M사이의 이항관계 $I \subseteq G \times M$로 구성된다. 어떤 객체 g가 속성 m을 가지고 있을 경우, gIm 또는 $(g, m) \in I$로 나타내며, g는 m을 갖는다는 것을 의미한다. <표 1>은 car reservation에 관한 context의 예이다.

context K = (G, M, I)에 대하여, $O \subseteq G$, $A \subseteq M$일 때, intent(O) = A ∧ extent(A) = O를 만족하는 (O, A)을 개념(formal concept)이라고 한다. 단, intent(O) := $\{a \in M | \forall o \in O : (o, a) \in I\}$, extent(A) := $\{o \in G | \forall a \in A : (o, a) \in I\}$. 임의의 $O \subseteq G$에 대하여, intent(O)에 의해 O의 모든 객체들이 공통적으로 갖는 속성들의 집합을 구할 수 있다. 예를 들면, 위의 <표 1>의 context에서, O = {Airport, Location, Downtown}에 대하여, intent(O) = {address}이다. 한편, 임의의 $A \subseteq M$에, extent(A)에 의해 A의 속성들을 갖는 객체들의 집합을 구할 수 있다. 예를 들면, A = {reservation_number, traveler}에 대하여, extent(A) = {Reservation}이다. 즉, 각 개념들은 (O, A)와 같은 형태의 쌍(pair)으로 정의되며 특히, 객체집합 O는 속

〈표 1〉 car reservation에 관한 context

	reservation_number	credit_card_number	payment_amount	drivers_license_number	name	model	address	traveler
Airport							×	
Location							×	
Driver				×	×			
Reservation	×							×
Check			×					
Payment_information			×					
Vehicle						×		
Credit_card		×	×					
Downtown							×	



[그림 4] 〈표 1〉로부터 추출된 개념격자

성집합 A의 extent이며, 동시에, 속성집합 A는 객체집합 O의 intent가 된다.

임의의 개념 (O_1, A_1), (O_2, A_2)에 대하여, $O_1 \subseteq O_2$ $(\Leftrightarrow A_1 \supseteq A_2)$라면 개념 (O_1, A_1), (O_2, A_2)은 상위-하위개념관계이며 $(O_1, A_1) \leq (O_2, A_2)$과 같이 표현한다. context K = (G, M, I)로부터 만들어진 모든 개념들 간의 상위 - 하위개념관계 ≤는 일종의 반순서관계에 해당하며, 개념들과 그들 사이의 상위-하위개념관계에 의해 만들어진 계층적 개념구조를 개념격자(Concept Lattice)라고 부르며, Hasse Diagram을 사용하여 가시화할 수 있다([그림 4] 참조).

개념격자를 나타낸 Hasse Diagram에서는, 각 개념들과 이들 사이의 상하위관계가 링크에 의해 표시되며, 특히, 개념들 간의 링크에 의해 만들어지는 경로에 의해 상위개념으로부터 하위개념으로 속성들이 상속되며, 하위개념으로부터 상위개념으로 해당 객체들이 전파된다. 예를 들어, Credit_card은 자신만의 고유속성인 credit_card_number를 가지며, 상위개념들로부터 payment_amount 속성을 상속받는다. 한편 payment_amount를 속성으로 갖는 객체로서는 Check, Payment_information, Credit_card가 됨을 알 수 있다. 이와 같은 방법을 사용함으로써, 주어진 문제영역의 객체들과 이들이 갖는 속성들을 context 형태로 파악하여, 개념을 추출하고 개념격자 형태로 나타냄으로써, 도메인 내의 개념들을 분류하고 체계화 할 수 있는 계층적 개념구조를 수월하게 구축할 수 있다.

또한, 형식개념분석 기법은 여러 가지 다양한 값을 가지는 속성들과 객체들, 그리고 객체와 속성 사이의 관계를 나타내는 데이터에도 적용될 수 있다. 예를 들어, “color”라는 속성은 “red”, “green” 또는 “white” 등과 같은 다양한 값들을 가질 수 있다. 이러한 속성들을 many-valued attributes라고 부른다.

many-valued attributes를 포함한 context를 many-valued context라고하며, 구체적으로, Many-valued context K = (G, M, W, I)는 객체들(Objects)의 집합 G와 속성들(Many-valued Attributes)의 집합 M, 속성의 값 W, 그리고 G와 M과 W 사이의 관계 $I \subseteq G \times M \times W$로 구성된다. 즉, G와 M의 원소들은 각각 해당 context의 객체들과 각 객체들이 가질 수 있는 속성들, 그리고 그 속성의 값들을 나타낸다. 또한, 어떤 객체 g가 속성 m을 가지고 있고 그 속성의 값이 w인 경우, $(g, m, w) \in I$ 또는 $m(g) = w$로 나타내며, 객체 g는 w 값을 가지는 속성 m을 갖는다는 것을 의미한다. Many-valued context도 One-valued context와 같이 테이블로 나타낼 수 있으며, 테이블의 각 셀에는 “X”표시 대신 해당 객체가 갖는 속성들의 값을 표시한다(<표 2> 참조).

이와 같은 Many-valued context로부터 개념들을 추출하고 개념격자를 구성하기위해서는 특정한 규칙에 따라 Many-valued context를 One-valued context로 변환할 필요가 있다. 이와 같이 변환된 one-valued context를 derived context라고 부르며 이러한 변환과정을 스케일링이라고 한다. 스케일링을 수행하기 위해서, Many-valued context의 각 속성들은 scale context를 토대로 해석되어 One-valued context로 변환된다. 예를 들어, <표 2>와 같은 Many-valued context의 속성들 중에서 Domain과 Range속성 각각에 대하여 <표 3>과 <표 4>의 scale context를 적용하여, 손실된 정보 없이

〈표 2〉 car reservation에 관한 many-valued context

	domain	range
payment	Driver	Payment_infor
traveler	Reservation	Driver
vehicle	Reservation	Vehicle
drop_off_location	Reservation	Location
pick_up_location	Reservation	Location

〈표 3〉 domain에 관한 scale context

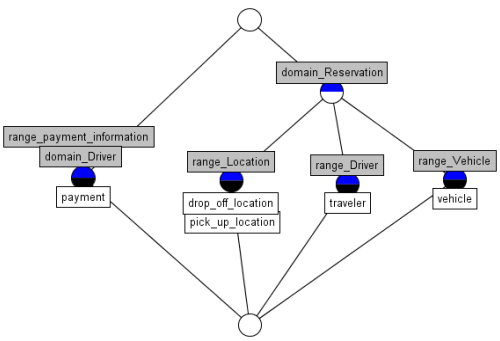
domain	Driver	Reservation
Driver	X	
Reservation		X

〈표 4〉 range에 관한 scale context

range	Payment_ information	Driver	Vehicle	Location
Payment_ information	X			
Driver		X		
Vehicle			X	
Location				X

〈표 5〉 〈표 2〉에 대한 Scaling 적용 결과

	domain		range			
	Driver	Reservation	payment_ information	Driver	Vehicle	Location
payment	X		X			
traveler		X		X		
vehicle		X			X	
drop_off_ location		X				X
pick_up_ location		X				X



[그림 5] 〈표 5〉의 개념격자

<표 5>와 같은 One-valued context로 변환 할 수 있으며, 이를 토대로 [그림 5]와 같은 개념계층구조를 구축할 수 있다.

3. 온톨로지의 오류검출

3.1 Ontology Context Model

온톨로지에 포함되어 있는 의미정보를 분석하기 위해서 형식개념분석기법을 응용하여, 온톨로지를 구성하는 각 요소들의 정보를 Context형태로 작성하여 입력해야 한다. 그러나 온톨로지의 정보는 매우 복잡하고, 방대하기 때문에 하나의 Context로 온톨로지 구성요소들의 다종다양한 정보를 적절하게 표현하기 어렵다. 따라서 본 연구에서는, 온톨로지의 핵심기본요소들을 표현할 수 있는 Ontology Context Model을 정의하였다.

Ontology Context Model은 총 3개의 One-valued Context와 2개의 Many-valued Context로 구성되어 있으며, 온톨로지의 핵심요소들(Class, Is-a관계, Property, Individual등)과 요소들 사이에 다양한 관계(Is-a관계, Instance-of, Has-property, Association) 및 관련정보(Property, individual의 상세정보)들을 표현하고 있다. Ontology Context Model의 정형화된 정의는 아래와 같다.

[정의1] Ontology Context Model $CF = \{S_K, B_J, I_J\}$는 온톨로지의 주요요소들의 집합인 S_K와 온톨로지의 요소들 사이의 이항관계들의 집합 B_J, 삼항관계 I_J로 구성된다.

- $S_K = \{ S_c, S_p, S_i, S_d, S_v, S_r \}$
  S_c : 온톨로지의 클래스들의 집합
  S_p : 온톨로지의 프로퍼티들의 집합
  S_i : 온톨로지의 individual들의 집합
  S_d : 온톨로지의 primitive data type들의 집합
  S_v : 온톨로지의 primitive data value들의 집합
  S_r = {domain, range}
- $B_J = \{ B_{cc}, B_{cp}, B_{ci} \}$
  $B_{cc} : \tau(B_{cc}) = (S_c, S_c)$, $B_{cc} \subseteq S_c \times S_c$
  (클래스들 사이의 Is-a관계)
  $B_{cp} : \tau(B_{cp}) = (S_c , S_p)$, $B_{cp} \subseteq S_c \times S_p$
  (클래스들과 프로퍼티들 사이의 Has-property 관계)
  $B_{ci} : \tau(B_{ci}) = (S_c, S_i)$, $B_{ci} \subseteq S_c \times S_i$
  (클래스들과 individual들 사이의 instance-of관계)
- $I_J = \{ I_{pr}, I_{ip} \}$
  $I_{pr} : \gamma(I_{pr}) = (S_p, S_r, S_c \cup S_d)$, $B_{pr} \subseteq S_p \times S_r \times$

($S_c \cup S_d$)
(프로퍼티들의domain, range정보)
$I_{ip} : \aleph(I_{ip}) = (S_i, S_p, S_i \cup S_v), B_{ip} \subseteq S_c \times S_p \times (S_i \cup S_v)$
(individual들의 실제 데이터정보)

단, $\tau(J) = (k1, k2)$ 함수는 이항관계 집합을 입력하여, 이항관계를 만족하는 k1, k2 집합을 출력하고, $\aleph(J) = (k1, k2, k3)$ 함수는 삼항관계 집합을 입력하여, 삼항관계를 만족하는 k1, k2, k3 집합을 출력한다.

위와 같은 온톨로지의 요소집합, 그들 사이의 이항관계 및 삼항관계 집합을 이용하여 온톨로지의 주요요소들과 그들 사이의 관계를 Ontology Context Model로 표현할 수 있다.

Ontology Context Model $CF = \{S_K, B_J, I_J\}$

- $S_K = \{ S_c, S_p, S_i, S_d, S_v, S_r \}$
  $S_c = \{$ Location, Airport, Downtown, Reservation $\}$
  $S_p = \{$ address, drop_off_location $\}$
  $S_i = \{$ airport_1, downtown_1 $\}$
  S_d : integer, string, boolean, float등 온톨로지 언어에서 지원하는 데이터타입의 집합
  S_v : 온톨로지 언어에서 지원하는 데이터타입의 실제 데이터들의 집합
  $S_r = \{$ domain, range $\}$
- $B_J = \{ B_{cc}, B_{cp}, B_{ci} \}$
  $B_{cc} = \{$(Location, Airport), (Location, Downtown) $\}$
  $B_{cp} = \{$ (Location, address), (Airport, address), (Downtown, address), (Reservation, drop_off_location) $\}$
  $B_{ci} = \{$ (Airport, airport_1), (Downtown, downtown_1)$\}$
- $I_J = \{ I_{pr}, I_{ip} \}$
  $I_{pr} = \{$ (address, domain, Location), (address, range, String), (drop_off_location, domain, Reservation), (drop_off_location, range, Location)$\}$
  $I_{ip} = \{$ (airport_1, address, Tokyo), (downtown_1, address, Seoul)$\}$

이와같은 Ontology Context Model을 기반으로, 본 연구에서는 방대하고 복잡한 온톨로지의 구성요소에 관한 정보를 입력으로 하고, 형식개념분석 기법을 적용하여 온톨로지에 내재되어 있는 개념구조를 파악함으로써, 온톨로지에 대한 개념분석을 수행한다.

3.2 온톨로지의 오류검출 알고리즘

본 연구에서 주목하고 있는 온톨로지에 존재할 수 있는 오류들 중에서 주요한 3가지 구조적 오류들(subClassOf 관계 오류, 프로퍼티 오류, individual의 오류)을 명확하게 정의한다. 이와 같은 정의들을 토대로, 온톨로지 소스코드를 Ontology Context Model로 변환하여 개념들을 추출하고 계층 구조화하여 분석함으로써, 온톨로지에 존재하는 구조적인 오류들을 수월하게 파악할 수 있다.

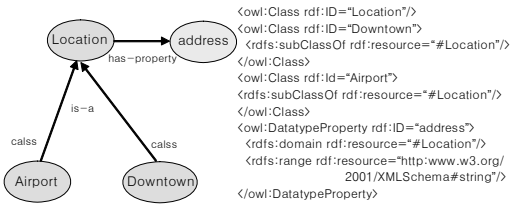
3.2.1 subClassOf 관계 오류

[정의 2] 임의의 온톨로지 O에 포함되는 클래스들의 집합 C의 임의의 $C_1, C_2 \in C$에 대해서, C_1이 C_2와 subClassOf 관계로 명시되어 있을 때(C_1이 C_2의 subClass), C_1이 갖는 프로퍼티집합 P_1과 C_2가 갖는 프로퍼티 집합 P_2가 서로 같다면 **C_1과 C_2사이에는 subClassOf 관계 오류가 존재한다** 라고 한다.

온톨로지에서 가장 중요하고, 핵심이 되는 요소는 클래스이다. 클래스는 클래스가 가지는 프로퍼티에 의해서 표현되어 진다. 따라서 특정 개념을 클래스로 표현한다면 개념이 가지고 있는 속성을 잘 파악하여, 클래스에 프로퍼티로 정의함으로써,

도메인의 정보가 제대로 반영된, 유용한 온톨로지가 될 수 있다. 그러나 이러한 기본적인 원칙을 간과한다면, 클래스는 물론, 클래스의 계층구조에도 문제가 발생할 가능성이 높다.

subClassOf 관계 오류란 온톨로지의 가장 핵심 요소인 클래스들 사이에 상-하위관계를 클래스가 갖는 속성을 고려하지 않고, 형식적으로만 subClassOf 관계를 명시한 상태를 말한다. 하위클래스는 상위클래스의 프로퍼티를 상속받고, 상위클래스와 구분이 되는 프로퍼티를 추가적으로 선언하거나, 프로퍼티의 range를 더 상위개념으로 선언함으로써 프로퍼티 중심적인 디자인관점에서 상 - 하위 클래스사이에 엄밀한 포섭관계(Subsumption)가 성립되어야 한다. 그러나 대부분의 온톨로지는 매우 복잡하고 사람에 의해서 수작업으로 구축되어지므로, subClassOf 관계 오류가 빈번하게 발생한다.



[그림 6] 온톨로지의 is-a 관계에 대한 예제

〈표 6〉 subClassOf 관계오류 검출알고리즘

```
// input : 온톨로지에서 subClassOf관계를 이용하여
   상하위관계가 정의된 2개의 클래스
// output : 문제가 없을시 true, 문제 발생시 false
Is-A_Check(superClass, subClass) {
 if (subClass.numberOfProperties
        ==superClass.numberOfProperties){
    while (numberOfProperties){
       if (subClass.Property.range
              ≥ superClassProperty.range)
          return false ;
       }
    }
 else if(subClass.numberOfPropertes
          < superClass.numberOfProperties)
       return false ;
 else
       return true ;
}
```

예를 들어 [그림 6]의 Location은 하위클래스로서 Airport, Downtown 클래스가 선언되어 있다. 그러나 Airport와 Downtown 클래스들은 Location의 하위클래스로서, Location과 구분이 되는 속성을 갖지 않기 때문에 subClassOf 관계 오류가 발생한다. subClassOf 관계 오류를 검출하기 위한 알고리즘은 <표 6>과 같다.

3.2.2 프로퍼티 오류

[정의 3] 임의의 온톨로지 O에 포함되는 클래스들의 집합C, 프로퍼티들의 집합P, O에서 지원하는 데이터 타입D가 주어졌을때, 임의의 $P_1 \in P$에 대해서, $domain(p_1) \notin C$이거나 $range(p_1) \notin (C \cup D)$라면, **$P_1$에는 프로퍼티 오류가 존재한다**라고 정의한다. 단, $domain(p_1)$과 $range(p_1)$은, 각각 p_1의 domain과 range이다.

온톨로지의 프로퍼티는 기본적으로 domain과 range정보를 수반한다. 따라서 프로퍼티를 정의하

〈표 7〉 프로퍼티오류 검출알고리즘

```
// input : 온톨로지에 정의되어진 프로퍼티
// output : 문제가 없을시 true, 문제 발생시 false
PropertyCheck(property) {
   boolean do == false ;
    boolean ran == false ;
    if ( property.domain == null ||
                          property.range == null ) {
         return false ;
   }
   while ( numberOfClass ) {
        if ( property.domain.type == class )
             do = true ;
        if ( property.range.type == class )
             ran = true ;
   }
   while ( numberOfDatatype ) {
        if ( property.range.type == datatype )
             ran = true ;
   }
   if ( do == false || ran == false )
        return false ;
   return true ;
}
```

기 위해서는 반드시 domain과 range 항목을 명확하게 기술해야 한다. domain에는 해당 프로퍼티가 선언되어진 온톨로지 또는 참조되어지는 온톨로지의 클래스가 기술되어야 하며, range에는 클래스 또는 온톨로지에서 지원하는 데이터타입을 선언해야 한다. 그러나 많은 온톨로지에서는 프로퍼티의 domain과 range 속성을 간과하고, 형식적으로만 프로퍼티를 선언하는 경우가 많다. 이와같은 문제점을 기반으로 프로퍼티오류를 검출하기 위한 알고리즘을 <표 7>과 같이 정의하였다.

3.2.3 Individual 오류

[정의 4] 임의의 온톨로지 O에 포함되는 Individual들의 집합을 I, 클래스들의 집합을 C라고 하자. 임의의 $I_1 \in I$, $C_1 \in C$에 대해서, I_1이 C_1과 instance-of 관계로 명시 되어 있고(I_1이 C_1의 instance), C_1이 갖는 속성집합이 P일 때, I_1이 $P_1 \in P$의 데이터 타입을 만족하지 않는 데이터 값을 갖는 경우, **I_1은 individual 오류가 존재한다라**고 정의한다.

온톨로지에 선언되어진 individual은 기본적으로 individual이 포함되는 클래스가 갖는 프로퍼티들의 range에 정의된 데이터타입의 데이터 값 또는 다른 individual을 참조해야 한다. 그러나 많은 온톨로지에서는 individual을 선언함에 있어서, individual이 속하는 클래스의 프로퍼티 타입에 적합하지 않는 데이터 값을 할당하여 individual의 오류를 발생시킨다. <표 8>의 알고리즘은 하나의 individual을 입력으로 individual에 속하는 클래스의 프로퍼티를 반영하여 실제 데이터 값을 갖는지 확인할 수 있다.

〈표 8〉 individual 오류 검출알고리즘

```
// input : 온톨로지에 정의되어진 individual
// output : 문제가 없을시 true, 문제 발생시 false
IndividualCheck(individual) {
   class = individual.getClass ;
   while ( class.numberOfProperties ) {
      if ( individual.property.value == null ||
           individual.property.type
                        !=property.range.type ) {
           return false ;
      }
   }
   return true ;
}
```

3.3 온톨로지 오류검출도구

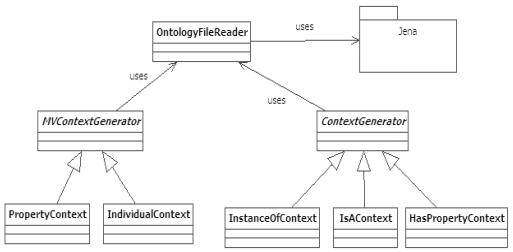
본 연구에서는 [그림 2]와 같이 4개의 모듈로 구성된 온톨로지 분석 및 오류검출도구를 개발하였다. 이 도구에서는, 입력된 온톨로지 소스코드로부터 개념분석을 수행하여 온톨로지에 내재된 에러들을 검출한다.

3.3.1 Context Family 생성 모듈

입력으로 주어진 온톨로지 소스코드로부터 Jena (http://jena.sourceforge.net)라는 온톨로지 추론도구를 사용하여 온톨로지의 주요요소(클래스, 프로퍼티, Individual)들 및 요소들 사이의 다양한 관계(Is-a relationship, instance-of relation ship, has-property relationship etc)들을 추출하고, 이 정보를 기반으로 5종류의 Context로 구성된 Ontology Context Model을 작성하는 모듈이다. 전처리 모듈은 [그림 7]과 같은 클래스들로 설계되어 있으며, 각각의 클래스들 기능은 <표 9>와 같다.

3.3.2 형식개념분석 모듈

제 2.2장에서 언급한 형식개념분석기법의 제반정의를 구현한 모듈로서, 전처리 모듈에서 생성된 Ontology Context Model을 입력으로 개념격자형태의 온톨로지 개념구조들을 구축하는 모듈이다. 형식개념분석 모듈에서 구축되어지는 개념격자를 자세히 분석함으로써, 온톨로지의 상세정보 및 구조적인 분석이 가능하다. 형식개념분석 모듈은 [그림 8]과 같은 클래스들로 구성되어 있으며, 각 클래스들에 대한 설



[그림 7] Context Family 생성 모듈의 클래스 설계도

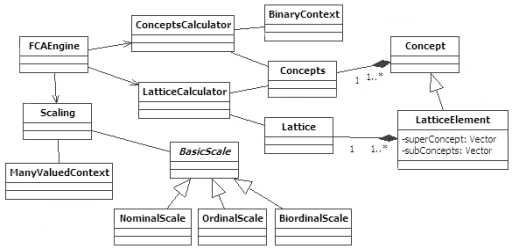
〈표 9〉 Context Family 생성모듈의 클래스들

클래스 이름	설 명
Ontology FileReader	Jena API를 이용하여 온톨로지의 정보를 조작하고 접근하기 위하여, 온톨로지 파일을 Jena 고유모델(OntoModel)로 객체화한다.
Context Generator	Ontology Context Model 요소들 사이의 One-valued Context를 생성하기위한 추상 클래스
MVContext Generator	Ontology Context Model 요소들 사이의 Many-valued Context를 생성하기위한 추상 클래스
IsAContext	OntoModel로 부터 클래스들 사이의 Is-a 정보를 One-valued Context로 생성
InstanceOf Context	OntoModel로 부터 클래스들과 Individual들 사이의 instance-of 정보를 One-valued Context로 생성
HasProperty Context	OntoModel로 부터 클래스들이 갖는 프로퍼티에 대한 정보를 One-valued Context로 생성
Property Context	OntoModel로 부터 프로퍼티들이 갖는 domain, range값을 Many-valued Context로 생성
Individual Context	OntoModel로 부터 individual들이 갖는 실제 데이터 값을 Many-valued Context로 생성

명은 <표 10>과 같다.

3.3.3 개념격자 가시화모듈

개념분석기법 모듈의 결과물인 개념격자들을 JPowerGraph(http://sourceforge.net/projects/jpowergraph/)라는 그래프라이브러리를 사용하여, 사용자에게 그래픽적인 표현을 제공한다. 개념격자 가시화모듈의 클래스 설계도는 [그림 9]와

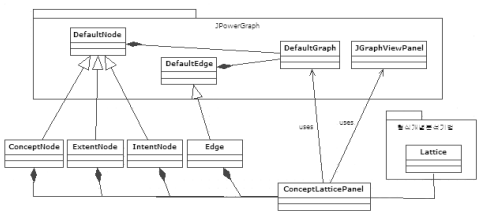


[그림 8] 형식개념분석 모듈의 클래스 설계도

〈표 10〉 형식개념분석 모듈의 클래스들

클래스 이름	설 명
FCAEngine	형식개념분석기법의 모든 기능을 관리하는 클래스
Concept Calculator	Context 클래스로부터 모든 Concept들을 구하는 클래스
Lattice Calculator	ConceptCalculator 클래스에서 구해진 Concept들을 토대로 Lattice 구조를 구하는 클래스
Scaing	Many-valued Context를 스케일링하는 과정을 구현한 클래스
BasicScale	NominalScale, OrdinalScale, BiordinalScale등 다양한 scaling 기법을 구현하기위한 추상클래스
Binary Context	Binary Context 정보를 표현한 클래스
ManyValued Context	Many-valued Context 정보를 표현한 클래스
Concept	Concept의 정보를 표현한 클래스
Lattice	Lattice의 정보를 표현한 클래스
Lattice Elements	Lattice의 요소들을 표현한 클래스로서 Concept 클래스로부터 상속받고 있다.

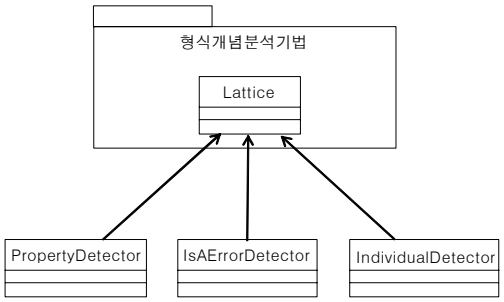
같으며, <표 11>은 각 클래스들의 기능을 설명하고 있다.



[그림 9] 개념격자 가시화모듈의 클래스 설계도

〈표 11〉 개념격자 가시화모듈의 클래스들

클래스이름	설 명
ConceptNode, Extent, IntentNode	화면에 그려질 Node의 속성정보(색상, 모양)를 표현하는 클래스
Edge	화면에 그려질 Edge의 속성정보를 표현하는 클래스
ConceptLattice Panel	Lattice 정보를 토대로 Node와 Edge를 생성하여 개념격자의 DefaultGraph에 입력하는 클래스
DefaultGraph and JGraph ViewPanel	JPowerGraph에 미리 구현된 클래스들로서, Node 정보와 Edge 정보를 입력받아서 자동으로 Graph를 그래픽 적으로 표현하고 Layout하는 클래스



[그림 10] 에러추출 모듈의 클래스 설계도

3.3.4 에러추출 모듈

에러추출 모듈은 개념격자로 표현된 온톨로지의 정보를 제 3.2장에서 정의한 온톨로지의 3가지 구조적인 오류를 자동으로 추출하는 모듈이다. [그림 10]에서 PropertyDetector 클래스는 프로퍼티의 오류, IsAErrorDetector 클래스는 subClassOf 관계 오류, 그리고 IndividualDetector 클래스는 Individual의 오류를 각각 검출하는 클래스이다.

4. 온톨로지 오류검출 실험

본 연구에서 개발한 온톨로지 분석도구를 사용하여 Prompt(http://protege.stanford.edu/plugins/prompt/prompt.html)의 튜토리얼 예제로 제공되는 Car reservation 온톨로지를 분석하고 구조적인 오류를 검출하는 사례를 설명한다.

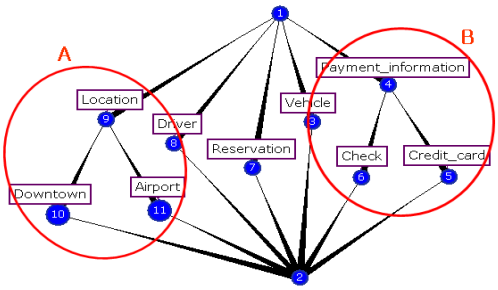
4.1 subClassOf 관계 오류

제 3.2장에서 언급한 온톨로지 subClassOf 관계 오류는 온톨로지의 Is-a 관계에 대한 개념구조를 나타내는 [그림 11]와 온톨로지의 has-property 관계에 대한 개념구조를 표현한 [그림 12]을 비교함으로써 문제를 파악할 수 있다. 클래스들 사이에 is-a 관계를 표현하고 있는 [그림 11]의 A부분에서 Location 클래스의 하위클래스로 Downtown과 Airport 클래스가 존재하지만, 클래스가 갖는 프로퍼티를 표현하고 있는 [그림 12]의 A'부분에서는 3개의 클래스(Location, Downtown, Airport)들이 모두 address프로퍼티만 정의하고 있기 때문에 같은 개념에 속하고 있다. 따라서 온톨로지 개발자가 의도한 대로 온톨로지의 구조가 성립하기 위해서, Airport, Downtown 클래스들에 각각의 클래스가 갖는 새로운 프로퍼티가 정의되어야 한다. 또한 [그림 11]의 B부분과 [그림 12]의 B'부분을 비교함으로써 subClassOf 관계 오류를 검출할 수 있다.

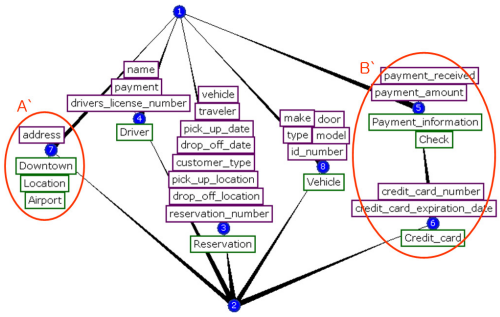
[그림 13]은 본 연구에서 개발된 도구를 사용해서 클래스계층구조를 자동으로 추출하여 가시화한 모습이다. 붉게 표시된 노드와 에지들이 현재 문제가 되고 있는 subClassOf 관계 오류를 나타낸다. 현재 car reservation 온톨로지에는 Location-Downtown, Location-Airport 그리고 Payment_information-Check 클래스사이에 is-a 관계가 엄밀한 포섭관계에 의해서 정의되지 않았음을 파악할 수 있다.

4.2 프로퍼티 오류

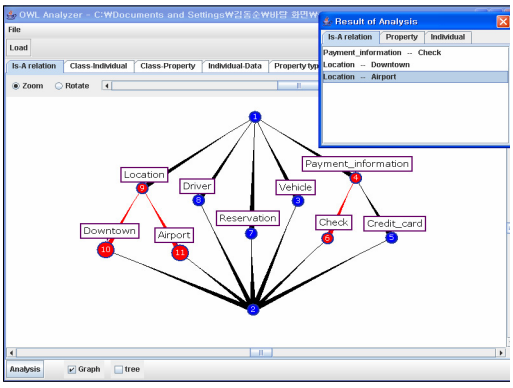
Protégé에서 제공하는 Pizza 온톨로지를 분석한 결과 중, 온톨로지의 프로퍼티의 개념구조를 표현한 개념격자를 분석하여 프로퍼티의 오류를 파악할 수 있었다. 프로퍼티의 개념격자에 포함된 개념들에서 intent 정보를 갖지 않는 개념들을 추출함으로써, 온톨로지에 프로퍼티 중에서 domain과



[그림11] Car reservation온톨로지의 is-a 관계에 대한 개념격자



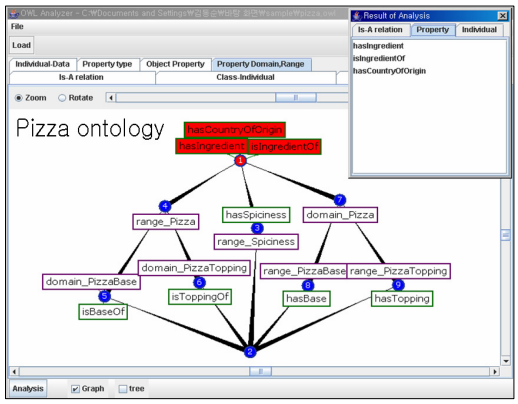
[그림 12] Car reservation 온톨로지의 has-proeprty에 대한 개념격자



[그림13] subClassOf 관계오류 검출

range 속성을 갖고 있지 않거나 비어있는 프로퍼티들을 추출할 수 있다.

[그림 14]는 Pizza 온톨로지의 프로퍼티에 관한 개념격자이다. 최상위개념은 intent 정보를 갖고 있지 않기 때문에, 최상위개념에 속하는 3개의 프로퍼티(hasCountryOfOrigin, hasIngredient, IsIngredientOf)들은 오류가 있음을 파악할 수 있다. 또한 프로퍼티들에 공통적인 특징을 파악할 수 있다. 예를 들어 hasTopping과 hasBase 프로퍼티들은 domain속성으로 Pizza 클래스가 공통적으로 포함되며, 또한 isBaseOf와 isToppingOf 프로퍼티들은 range 속성으로 Pizza 클래스가 공통적으로 포함된다.



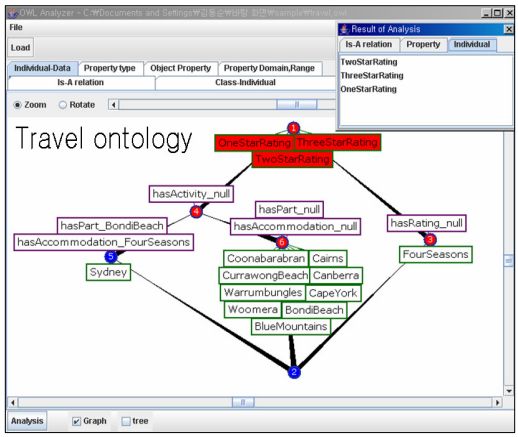
[그림 14] 프로퍼티오류 검출

4.3 Individual 오류

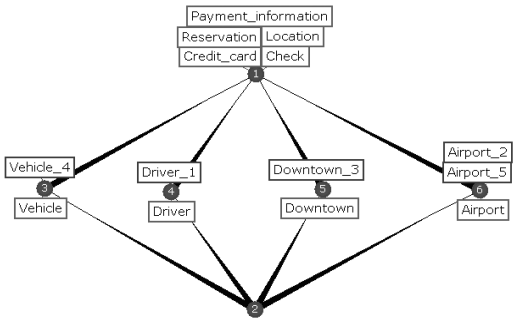
Travel 온톨로지를 분석한 결과 중, 온톨로지의 individual의 개념구조를 표현한 개념격자를 분석하여 individual의 오류를 파악할 수 있었다. individual들의 개념격자에 포함된 개념들에서 intent정보를 갖지 않는 개념들을 추출함으로써, 온톨로지에 정의된 individual중 해당 클래스의 프로퍼티에 적합한 데이터 값을 제대로 정의하지 않는 individual들을 검출할 수 있다. [그림 15]의 Travel 온톨로지에는 3개의 individual(OneStarRating, ThreeStarRating, TwoStarRating)들에 오류가 있음을 파악할 수 있다.

앞서 설명한 온톨로지의 구조적 오류를 파악하기 위해 사용된 개념구조뿐만 아니라, 사용되지 않은 개념구조들도 온톨로지의 의미 있는 정보를 표현하고 있다. Instance-Of 관계를 표현한 개념격

자를 이용하여 온톨로지의 모든 individual들을 해당 클래스별로 클러스터링한 정보를 파악할 수 있다. 예를 들어 [그림 16]은 Car Reservation 온톨로지의 instance-of 관계를 표현한 개념격자로서, Airport에 속하는 individual(Airport_2, Airport_5)들, 그리고 Vehicle, Driver, Downtown클래스에 각각 한 개의 individual이 속해 있음을 파악할 수 있다.

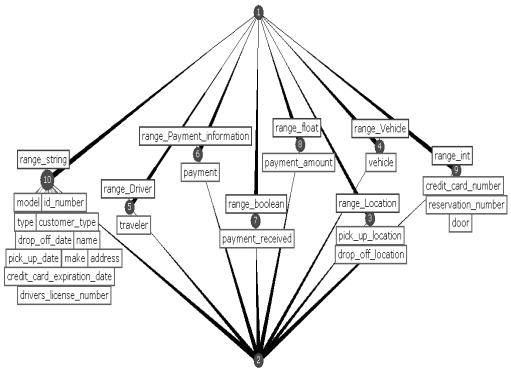


[그림 15] Individual 오류 검출



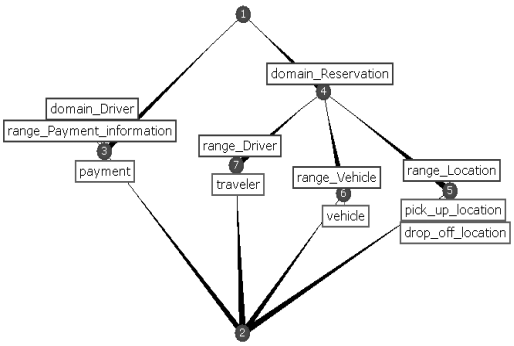
[그림 16] Car Reservation 온톨로지의 instance-of에 대한 개념격자

한편, 프로퍼티를 표현한 Many-valued Context로부터 Range 속성만을 스케일링하여 구축된 개념격자는, 프로퍼티를 데이터 타입별로 분류하여 분석해주고 있다. 예를 들어 [그림 17]은 Car Reservation 온톨로지의 프로퍼티들을 데이터 타입별로 분석한 개념격자이다. int형 타입의 프로퍼티는 credit_card_number, reservation_number, door임을 알 수 있으며, 그 외에도 Car Reservation 온톨로지에는 4가지 클래스(Driver, Location, Vehicle, Payment_information)타입과 4가지 primitive데이터 타입(string, boolean, float, int)들의 프로퍼티들이 정의되어 있음을 파악할 수 있다.



[그림 17] Car Reservation 온톨로지의 프로퍼티 타입에 대한 개념격자

Object프로퍼티의 domain과 range 속성을 스케일링한 개념구조는 클래스들 사이에 association 관계를 표현하고 있다. [그림 18]은 Car Reservation 온톨로지의 Object 프로퍼터에 대한 개념격자이다. 예를 들어 Payment_information 클래스와 Driver 클래스는 payment라는 object 프로퍼티에 의해 연관



[그림 18] Car Reservation 온톨로지의 Object 프로퍼티에 대한 개념격자

관계를 맺고 있다. 또한, Location, Driver, Vehicle 클래스들은 공통적으로 Reservation 클래스와 연관관계가 존재함을 파악할 수 있다.

이상의 분석사례와 더불어서, 본 논문에서 개발한 온톨로지 분석 및 오류추출도구를 이용하여, 다양한 온톨로지 구축도구들(Protégé, Hozo, ez-owl, KAON)[1]에서 제공한 총 18개의 온톨로지 예제들을 대상으로 개념계층구조를 기반으로 하는 분석과 오류검사를 실시하였다. Protégé에서 제공한 7개의 온톨로지중에서 subClassOf 관계 오류는 7개의 온톨로지에서, 프로퍼티 오류는 3개, individual의 오류는 3개의 온톨로지에서 검출되었다. 또한 Hozo에서 제공한 5개의 온톨로지 중 subClassOf 관계 오류는 4개, 프로퍼티 오류는 4개가 검출되었으며 individual의 오류는 검출되지 않았다(<표 12> 참조). 본 실험결과로부터, 각종 유명한 온톨로지 개발도구에 포함된 예제 온톨로지에서도 조차 여러 가지 오류들이 포함되어 있음을 알 수 있었으며, 이와 같은 오류들은, 본 연구에서 개발한 온톨로지 분석 및 오류검출도구를 사용하여 수월하게 분석/추출할 수 있고, 온톨로지 개발자에게 보다 좋은 개념계층구조를 갖는 온톨로지를 개발할 수 있는 유용한 정보를 제공해 준다.

〈표 12〉 온톨로지 오류검출 실험결과

온톨로지 제공도구	Protégé	Hozo	ez-owl	KAON
실험온톨로지 갯수	7개	4개	3개	4개
subClassOf 관계 오류	7개	4개	3개	4개
프로퍼티 오류	3개	4개	3개	3개
individaul 오류	3개	0개	1개	3개

1) Protégé(http://protege.stanford.edu/), Hozo(http://www.ei.sanken.osaka-u.ac.jp/hozo/eng/index_en.php), ez-owl(http://iweb.etri.re.kr/ezowl), KAON (http://kaon.semanticweb.org).

5. 결 론

최근에는 온톨로지의 개발 및 관리에 관한 다양한 연구들이 진행되고 있다. 그러나, 대부분의 온톨로지 개발에서는 정형화된 온톨로지 개발방법론을 적용하지 않고, 개발자들의 경험에만 의존하여 온톨로지를 수작업으로 구축하고 있으며, 구축되는 대부분의 온톨로지들은 방대한 용량의 도메인 정보를 대상으로 하기 때문에 잘못된 정보를 포함하거나, 도메인의 정보를 제대로 반영하지 못하고 있을 가능성이 매우 높다. 따라서 온톨로지에 대한 적절한 분석 및 오류 검출에 관한 제반 이론 및 지원도구가 절대적으로 필요하다.

본 논문에서는 주어진 데이터로부터 개념구조를 중심으로 데이터를 분석하는 형식개념분석기법을 토대로, 온톨로지의 소스코드로부터 핵심요소들을 추출, 분석하기 위한 알고리즘을 제안하고 지원도구를 개발하였다. 구체적으로는, 온톨로지 소스코드로부터 온톨로지의 핵심구성요소들을 추출하는 공정과, 이를 형식개념분석기법에 적용시키기 위한 데이터모델을 정의하고, 해당 온톨로지의 개념구조들을 구축하기 위한 제반기법을 제안하고 몇 가지 예제 온톨로지를 대상으로 오류검출실험을 실시하여 본 연구결과의 유용성을 살펴보았다. 본 연구에서 개발된 온톨로지 오류검출기법은 온톨로지의 개발 및 구현 단계에서 뿐만 아니라, 구조적인 오류를 찾아내고 수정/보완하는 온톨로지 제반 관리활동에 있어서 유용한 기능을 제공한다.

참 고 문 헌

[1] Berners-Lee, T., J. Hendler, and O. Lassila, “The Semantic Web”, *Scientific American*, Vol.243, No.5(2001), pp.34-43.

[2] Cardoso, J., “The Semantic Web Vision : Where Are We?”, *IEEE Intelligent Systems*, Vol.22, No.5(2007), pp.84-88.

[3] Carpineto, C. and G. Romano, Concept Data

Analysis : Theory and Applications, Wiley, West Sussex, England, 2004.

[4] Davies, J., D. Fensel, and F. van Harmelen, eds, *Towards the Semantic Web : Ontology-driven Knowledge Management*, John Wiley and Sons, New York, 2003.

[5] Ganter, B. and R. Wille, *Formal Concept Analysis : Mathematical Foundations*, Springer-Verlag, Heidelberg, 1999.

[6] Gomez-Perez, A. and O. Corcho, "Ontology languages for the Semantic Web", *IEEE Intelligent Systems*, Vol.17, No.1(2002), pp.54-60.

[7] Julio, C. Arpírez, Ó. Corcho, and M. Fernández-López, A.Gómez-Pérez, "WebODE in a Nutshell", *AI Magazine*, Vol.24, No.3(2003), pp.37-47.

[8] OWL_Overview : http://www.w3.org/TR/owl-features/.

[9] Protégé : http://protege.stanford.edu/.

[10] RDF_Primer : http://www.w3.org/TR/rdf-primer/.

[11] Simperl E. and C. Tempich, "Ontology Engineering : A Reality Check", OTM Conferences, *LNCS* 4275(2006), pp.836-854.

[12] Sure Y., M. Erdmann, and R. Studer, *On-To-Knowledge : Semantic Web enabled Knowledge Management*, Wiley, New York, 2002.

◈ 저 자 소 개 ◈

황 석 형 (shwang@sunmoon.ac.kr)

1991년 강원대학교 전자계산학과를 조기졸업(이학사)하고, 일본 오사카대학교 대학원 정보공학과에서 객체지향소프트웨어공학 전공으로 공학석사(1994년)와 공학박사(1997년) 학위를 취득했다. 현재, 선문대학교 컴퓨터공학부 부교수로 재직 중이며, 주요 관심분야로는, 객체지향소프트웨어공학, Ontology Engineering, Formal Concept Analysis, Semantic Web 분야 등이며, IEICE, 정보처리학회 논문지 등에 다수의 논문을 실었다.