후방향 전진 추론을 이용한 RDF 모델의 효율적인 변경 탐지

(Efficient Change Detection between RDF Models Using Backward Chaining Strategy)

임 동 혁 † 김 형 주 ††

(Dong-Hyuk Im) (Hyoung-Joo Kim)

요 약 RDF(Resource Description Framework)는 시맨틱 웹에서 메타 정보를 기술하는 온톨로지 언어로 많이 사용되고 있다. 온톨로지는 실세계에 대한 모델링을 기반으로 하기 때문에 끊임없이 갱신이 발생한다. 이런 갱신을 찾고 분석하는 일은 지식 관리 시스템에서 핵심이 된다. 기존의 RDF 모델에 대한 변경 탐지 기법들은 구조적 변경에 초점을 두었으나 RDFS 함의 규칙을 적용하여 좀 더 작은 크기의 변경 부분을 찾는 연구들이 소개되고 있다. 하지만 RDF 모델의 추론은 데이타 크기와 시간의 증가에 영향을 미친다. 본 논문에서는 RDFS 함의 규칙을 효율적으로 사용하는 변경 탐지 기법을 제안한다. 제안된 기법은 후방향 전진 추론 기반으로 모델 일부분에만 추론을 적용하여 변경 내용을 계산한다. 실제 사용하는 RDF 데이타들을 사용하여 기존의 변경 탐지 기법과의 비교 실험을 통해 성능을 향상시킬 수 있음을 보인다.

키워드 : RDF, 온톨로지, 변경탐지, RDFS 함의 규칙, 추론

Abstract RDF is widely used as the ontology language for representing metadata on the semantic web. Since ontology models the real-world, ontology changes overtime. Thus, it is very important to detect and analyze changes in knowledge base system. Earlier studies on detecting changes between RDF models focused on the structural differences. Some techniques which reduce the size of the delta by considering the RDFS entailment rules have been introduced. However, inferencing with RDF models increases data size and upload time. In this paper, we propose a new change detection using RDF reasoning that only computes a small part of the implied triples using backward chaining strategy. We show that our approach efficiently detects changes through experiments with real-life RDF datasets.

Key words : RDF, Ontology, Change detection, RDFS entailment rules, Inference

1. 서 론

RDF(Resource Description Framework)는 웹 상에서 메타 정보를 표현하는 언어로 W3C에 의해 제정되었다[1]. RDF와 같은 온톨로지는 다양한 지식 도메인(시맨틱 웹, E-Commerce, 지식 관리)에서 많이 사용이 되는데 지식 도메인은 계속해서 진화하는 특성을 가지므로 RDF 데이타의 변경 부분을 찾는 것은 매우 중요하며 많은 연구가 되고 있다[2,3].

RDF는 주어(Subject), 술어(Predicate), 목적어(Object)의 트리플 구조를 갖는 트리플(문장)들로 이루어져 있으며 그래프 모델로 나타낼 수 있다[4]. 그림 1은 RDF 모델의 갱신 전후를 보여주고 있다. 그림 1에서처럼 각 RDF 모델은 트리플의 집합으로 표현이 가능하며 두 모

· 본 연구는 07 첨단도시A01, BK-21 정보기술 사업단과 지식 경제부 및 정보통신연구진흥원의 대학 IT연구센터 육성지원사업(IITA-2008-C1090-0801-0031)의 연구 결과로 수행되었음

† 학생회원 : 서울대학교 컴퓨터공학과
dhim@idb.snu.ac.kr
†† 종신회원 : 서울대학교 컴퓨터공학 교수
hjk@snu.ac.kr

논문접수 : 2008년 9월 10일
심사완료 : 2008년 12월 12일

정보과학회논문지: 컴퓨팅의 실제 및 레터 제15권 제2호(2009.2)

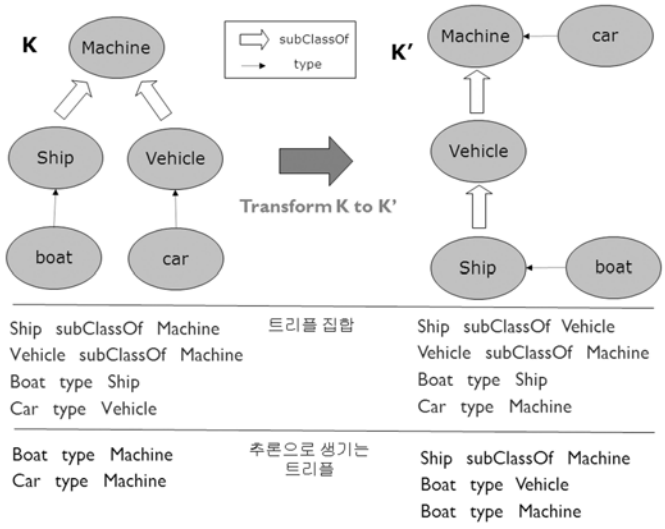


그림 1 RDF 모델의 갱신 전후

델의 변경 부분은 두 트리플 집합의 차이로 계산이 가능하다. 또한 RDF 모델에서는 추론을 통해 나타나지 않은 관계를 알아낼 수 있다. 이런 추론을 적용한 변경 탐지 기법들[5,6]은 좀 더 작은 크기의 변경 부분을 구할 수 있다. 예를 들면 그림 1에서 단순한 두 RDF 모델의 트리플들의 변경 부분을 계산하면 두 트리플 집합의 차집합을 계산하면 된다. 하지만 추론을 이용한 변경 탐지 기법을 사용하면 그림 1에서처럼 추론으로 인한 추가되는 트리플들이 생겨서 변경 된 내용을 줄일 수 있게 된다. 하지만 이런 추론은 13개의 RDFS 함의 규칙을 전방향 추론으로 미리 계산을 해야 한다. 즉 더 이상 추론된 문장이 없을 때까지 계속해서 추론을 하며 변경 탐지에 불필요한 규칙이 적용될 수도 있고 모델의 크기가 커짐에 따라 비효율성의 문제점이 생기게 된다.

이러한 문제점을 해결하기 위해 본 논문에서는 효율적인 변경 탐지 기법을 제안한다. 미리 계산하는 전방향 추론이 아닌 후방향 추론을 사용하며 RDF 모델 전체를 추론하는 것이 아니라 변경 탐지에 사용될 수 있는 부분들만을 추론하는 방법을 제안한다. 또한 기존의 다른 연구에서 수행되었던 변경 탐지 기법과의 비교 분석을 통해 제안하는 기법의 성능을 살펴보고, 이를 통해 적절한 추론 방식이 RDF 모델의 변경 탐지의 정확성과 성능을 향상 시킬 수 있음을 보인다.

본 논문의 구성은 다음과 같다. 2장에서는 관련 연구로 RDF 분야에서 연구 되었던 변경 탐지 기법의 이슈에 관한 연구들과 기존의 RDF 모델에서의 추론 방법에 대해 살펴본다. 3장에서는 데이타 모델 및 본문에서 사용하는 몇 가지 기본 개념을 정의하고, 기존의 변경 탐지 기법들과 문제점을 소개한다. 4장에서는 제안하는 변경 탐지 기법에 대해 설명하고, 5장에서는 실험을 통해 제안된 방법의 우수성을 보이고 6장에서는 결론을 내는 것으로 마무리 한다.

2. 관련연구

기존의 RDF에 대한 변경 탐지 기법은 구조적인 변경 부분에 대한 탐지에 초점을 두고 있다. [7]의 Canonicalization 기법은 RDF 문서를 암호화 할 때 사용되는 기법이나 두 개의 RDF 문서에서의 변경 탐지에도 사용이 된다. 두 RDF 그래프를 유일한 표준형으로 변환한 다음 두 표준형을 비교하여 두 RDF 모델의 차이를 구하고 있다. [8]은 두 RDF 그래프의 변경 부분을 탐지하기 위해 RDF 그래프 내부에서의 노드간의 의존적 관계를 이용하였다. 또한 변경 부분을 계산하는 것뿐만이 아니라 갱신 정보의 배포, 동기화에 대해 제안을 하고 있다. [9]에서는 공노드를 포함하는 RDF 모델의 변경 탐지 기법을 제시하였다. 공노드란 RDF 모델에서의 제한 요소를 기술할 때 사용되는 노드로써 두 RDF 모델의 비교시 같은 공노드인지를 비교해야만 하는 어려움을 가지고 있다. [9]에서는 이러한 공노드에 대한 비교 방법을 제시하고 있다.

온톨로지에서의 변경 탐지 연구는 주로 온톨로지의 개념간의 차이를 구하고 있다. [3]에서는 온톨로지의 버전이 다르더라도 최대한 호환되어 사용이 가능하도록 두 온톨로지 버전의 어떻게 다른지 사용자에게 보여주고 변화한 온톨로지의 개념들간의 관계를 사용자가 지정할 수 있도록 하고 있으며 [10]에서는 온톨로지의 두

버전간의 차이점을 휴리스틱 방법을 사용하여 사용자에게 알려주며 [11]에서는 그래프 비교를 통한 온톨로지 변경을 탐지한다.

RDF 모델의 추론에 대한 연구에 있어 [12]는 Sesame에서의 전방향 추론 방식이 질의 처리 시 매우 유용함을 보였다. 현재의 사실들에 RDFS 함의 규칙에 따라 미리 저장하여 질의 처리 시 사용하는 전략을 취하고 있다. 또한 가장 효율적인 전방향 알고리즘인 Rete 알고리즘[13]에서는 추론 규칙들간의 관계를 고려하여 추론을 하며 부분 매칭이 일어나는 규칙에는 중간 결과를 저장하여 중복된 추론 과정을 제거하여 추론 성능을 향상시키고 있다. 또한 [14]에서는 전방향 추론으로 인한 저장 크기를 줄이기 위해 몇 가지 추론 규칙을 저장 시간에 계산하지 않고 실행 시간에 추론하는 전략을 제안하고 있다. [15]에서는 여러 RDF 혹은 OWL 등의 온톨로지 저장 시스템에서의 추론 성능에 대한 비교 실험을 하여 추론 성능 향상의 필요성을 강조하고 있다.

3. 기본 개념

이 장에서는 본 논문에서 사용하는 RDF의 데이타 모델과 몇 가지 기본 개념들에 대해 설명한다.

3.1 RDF 데이타 모델과 추론 정의

RDF 모델은 DAG(Directed Acyclic Graph)로 표현되며 노드는 각 자원을 표현하고 간선은 노드 사이의 관계를 표현한다. RDF 그래프는 두 자원간의 이진 관계를 표현하는 트리플의 집합이며 트리플은 주어(Subject), 술어(Predicate), 목적어(Object)로 이루어진다. 또한 W3C의 권고안인 RDF Semantics의 함의 규칙[16]을 이용하여 추론을 수행할 수 있다. 추론은 기본적으로 기존의 트리플로부터 다른 트리플을 추가하는 구조를 갖는다. 그림 2는 RDF 모델의 함의 규칙을 나타낸다. 예를 들면 규칙 11의 경우 subClassOf 관계의 추이적 관계를 보여주고 있다. 자원 U가 자원 V의 하위클래스 관계이고 자원 V가 자원 X의 하위클래스 관계이면 자동적으로 자원 U가 자원 X의 하위클래스 관계를 갖는 것을 의미한다. 이 외에도 확장 함의 규칙, 데이타 타입 함의 규칙을 제공하고 있다. 하지만 본 논문에서는 추가적인 함의 규칙에 대해서는 고려하지 않는다.

3.2 RDF 모델에서의 변경 탐지 기법

본 절에서는 기존의 RDF 모델의 변경탐지 기법 △E, △C, △D에 대해 설명한다. K, K'는 RDF 모델을 나타내며 t는 RDF 모델에 속해있는 트리플을 표시하며 C(K)는 RDF 모델 K를 RDFS 함의 규칙에 따라 추론된 RDF 모델을 나타낸다.

정의 1. (Explicit Delta, △E) △E(K, K')는 다음과 같이 정의된다.

$$\triangle E(K, K') = \{Insert(t) \mid t \in K' - K\} \cup \{Delete(t) \mid t \in K - K'\}$$

정의 2. (Closure Delta, △C) △C(K, K')는 다음과 같이 정의된다.

$$\triangle C(K, K') = \{Insert(t) \mid t \in C(K') - C(K)\} \cup \{Delete(t) \mid t \in C(K) - C(K')\}$$

정의 3. (Dense Delta, △D) △D(K, K')는 다음과 같이 정의된다.

규칙	전제	결론
(1)	X A Y	A rdf:type rdf:Property
(2)	A rdfs:domain X U A Y	U rdf:type X
(3)	A rdfs:range X Y A V	V rdf:type X
(4)	U A B B A U	U rdf:type rdfs:Resource U rdf:type rdfs:Resource
(5)	U rdfs:subPropertyOf V V rdfs:subPropertyOf X	U rdfs:subPropertyOf X
(6)	U rdf:type rdf:Property	U rdfs:subProperyOf U
(7)	A rdfs:subPropertyOf B U A Y	U B Y
(8)	U rdf:type rdfs:Class	U rdfs:subClassOf rdfs:Resource
(9)	U rdfs:subClassOf X V rdf:type U	V rdf:type X
(10)	U rdf:type rdfs:Class	U rdfs:subClassOf U
(11)	U rdfs:subClassOf V V rdfs:subClassOf X	U rdfs:subClassOf X
(12)	U rdf:type rdfs:ContainermembershipProperty	U rdfs:subPropertyOf rdfs:member
(13)	U rdf:type rdfs:Datatype	U rdfs:subClassOf rdfs:Literal

그림 2 RDF 모델의 함의 규칙

$$\triangle D(K, K') = \{Insert(t) \mid t \in K' - C(K)\} \cup \{Delete(t) \mid t \in K - C(K')\}$$

예제 1. 그림 1의 두 RDF 모델 K, K'에서 위의 정의된 변경 탐지 기법을 사용한 결과는 다음과 같다.

△E(K, K') = { Insert(Ship subClassOf Vehicle), Insert(car type Machine),
Delete(Ship subClassOf Machine), Delete(car type Vehicle)}

△C(K, K') = { Insert(Ship subClassOf Vehicle), Insert(boat type Vehicle),
Delete(car type Vehicle)}

△D(K, K') = { Insert(Ship subClassOf Vehicle),
Delete(car type Vehicle)}

△E는 추론을 하지 않은 상태에서의 두 RDF 모델의 트리플 차집합을 계산하며 △C는 두 모델을 추론 시킨 후에 차집합을 계산한다. 반면에 △D는 추론되지 않은 모델에서 추론된 모델의 차집합을 각각 계산하게 된다. 위의 결과와 같이 함의 규칙을 적용함으로써 변경 탐지 결과가 다르게 나오는 것을 알 수 있다. 대다수의 변경 탐지 기법[3,7-9,11]은 RDF 그래프에서의 구조적 변경을 계산하는 △E에 속하며 SemVersion[5]에서는 △C을 사용하며 △D는 Delta Function[6]에서 사용하고 있다.

3.3 변경 탐지 기법의 한계

RDFS의 추론 전략은 추론 방식에 따라 전방향 추론(Forward chaining)과 후방향 추론(Backward chaining)으로 구분된다. 전방향 추론은 데이타를 로딩할 때 미리 추론을 계산한다. 따라서 데이타 로딩 시간이 증가하며 이에 대한 공간 비용을 많이 차지하게 된다. 전방향 추론의 장점은 미리 모든 정보를 저장하였기 때문에 빠른 질의 응답을 보인다는 것이다. 이에 반해 후방향 추론은 실행 시간에 추론을 하게 되므로 로딩 시간이 짧은 반면에 질의 응답 시간이 길게 된다[12].

대부분의 RDF 저장 시스템에서는 전방향 추론 방식을 많이 사용하고 있으며 앞에서 설명한 △E, △C, △D를 구하는 변경 탐지 기법 역시 13개의 RDFS 함의 규칙을 반복적으로 적용하여 더 이상 추론 된 문장이 없을 때 추론이 종료되는 전방향 추론을 사용하고 있다. 추론 이후에는 $t \in K'$ 만족하는 트리플 t에 대해 $t \notin K$인 경우에 $t \in C(K)$를 검사하여 t가 추론을 통해 변경 부분에 포함되는지를 검사하는 방법을 사용한다. 다시 말해 추론을 해서 더 줄일 수 있는 변경 부분을 찾는 것을 수행한다. 하지만 RDF 모델을 함의 규칙에 의해 추론한 후에 비교를 하면 변경 부분을 더 줄일 수도 있지만 추가적인 비용이 들게 된다. 표 1은 Uniprot RDF 데이타와 Gene Ontology RDF TermDB의 추론 결과를 보여준다. 표 1에서 보듯이 추론을 함으로써 시간적인 비용과 크기의 증가를 알 수 있다. 변경 부분을 계산하기 위해서는 두 모델의 트리플 집합을 비교하여야 하기 때문에 늘어난 크기만큼 비교를 수행해야 한다. [6]에서는 [17]과 같은 레이블링 기법을 이용하여 효율적으로 변경 탐지가 가능하다고 언급하고 있다. 다시 말해 $t \in C(K)$ 검사를 O(1)에 결정할 수 있다. 하지만 이런 접근에는 레이블을 구축하기 위한 전처리 단계를 요구하고 있으므로 빠르게 변경 탐지를 하기에는 적합하지 않다. 또한 클래스의 상하위 관계만을 처리할 수 있고 type 검사나 subPropertyOf와 같은 추론에는 사용할 수 없다. 따라서 대용량의 RDF 모델인 경우 효율성을 고려해야 한다.

표 1 RDF 데이타에 대한 추론 결과

데이타 \ 속성	크기	트리플 수	추론 후 추가된 트리플 수	추론 시간
UniProt Taxonomy (2008/2)	182MB	2,637,046	7,111,072	257(S)
Gene Ontology (2008/01)	32MB	409,671	376,807	11(S)

또한 추론을 사용하는 △C의 경우 의도하지 않은 변경 탐지 결과가 나올 수 있다. 예를 들면 예제 1에서의 △C 변경 탐지 결과에서 Insert(boat type Vehicle)는 원래의 구조적 변경△E에서는 나오지 않는 변경 탐지 결과이다. 즉 추론된 모델을 사용하기 때문에 변경 탐지 결과가 추론하기 전보다 더 커질 수도 있는 현상이 일어나게 된다. △D의 경우 추론된 두 모델간의 변경 부분을 계산하진 않더라도 변경 부분에 포함되지 않는 부분에 대해 추론을 하는 문제점을 가지고 있다. 이에 4장에서는 △C, △D의 장점인 RDFS 함의 규칙의 추론을 이용하여 변경 부분의 크기를 줄일 수 있으면서도 위와 같은 문제점에 유연하게 동작할 수 있는 변경 탐지 기법을 제안한다.

4. 후방향 전진 추론을 이용한 변경 탐지 기법

4장에서는 본 논문에서 제안하고 있는 변경 탐지 기법에 대해 설명한다.

4.1 변경 탐지 기법의 개관

본 논문에서 제안하는 변경 탐지 기법은 기존의 추론을 사용해서 변경 탐지를 구하는 방법에서 효율성을 고려한다. 기존의 변경 탐지 기법(△C, △D)이 미리 추론을 한 RDF 모델을 이용하는 반면에 제안된 방법은 먼저 변경 부분을 찾은 후에 변경 된 부분에 대해서만 추론을 하는 것이 기본적인 접근 방법이다. 그림 3은 기존의 방법과 제안된 방법과의 비교를 보여주고 있다.

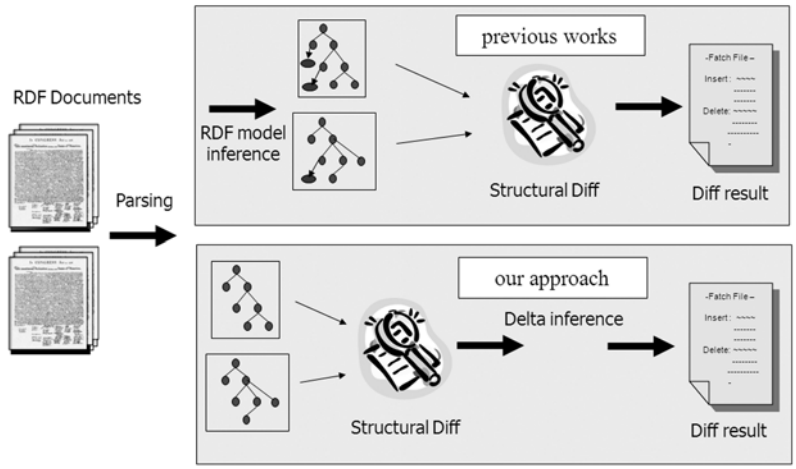


그림 3 기존 변경 탐지 기법과의 비교

4.2 Lazy & Closure 변경 탐지 기법

본 논문에서 제안하는 Lazy & Closure 변경 탐지 기법은 추론에 있어 후방향 전진 추론을 사용하며 특히 모든 트리플에 대해 추론 규칙을 적용하지 않는다. 먼저 트리플에서 구조적인 차집합을 구하고 계산된 변경 부분에 대해서 지연 추론을 하게 된다. 따라서 전체 RDF 모델이 아닌 변경된 내용에 대해서 RDFS 함의 규칙을 적용한다. 변경 탐지에서 추론이 사용되는 예를 살펴보면 한 RDF 모델의 트리플이 다른 모델의 트리플에 구조적으로 나타나지는 않지만 추론을 하게 되면 대응될 수 있는 경우이다. 그림 1의 예를 들면 모델 K'의 (Car type Machine) 트리플은 K 모델에서는 나타나지 않는다. 하지만 Vehicle이 Machine의 하위클래스 이므로 RDFS 함의 규칙 9번에 의해 (Car type Vehicle)에서 (Car type Machine)을 유도할 수 있다. 따라서 (Car type Machine) 트리플은 삽입되었다고 볼 수 없는 것이다. 이와 같은 RDFS 함의 규칙의 의미를 살펴보면 리소스의 타입 추론과 클래스, 술어의 계층 관계 생성과 그 계층 관계를 이용한 추론으로 구분 할 수 있다[15]. 이 중 변경 탐지에 사용될 수 있는 추론은 타입 추론 (규칙 9), 계층 관계 생성 (규칙 5, 규칙 11), 계층 관계를 이용한 규칙(규칙 7)이다. 나머지 규칙들은 RDFS의 기본 어휘의 확장들이므로 변경 탐지에 사용하는 것은 의미가 없다. 아래 정리는 변경 탐지에서 사용되는 추론의 특성을 설명한다.

정리 1. 추론을 통해 대응될 수 있는 트리플은 반드시 주어(Subject)가 같다.

추론을 통해 다른 모델의 트리플과 대응되는 트리플은 위에서 언급된 함의 규칙(규칙 5,7,9,11)의 적용을 받는다. 위의 규칙들은 추론을 통해 결론으로 추가되는 트리플들이 전제로 사용되었던 트리플과 주어가 같은 특성을 가지고 있는 것을 그림 2를 통해 확인할 수 있다.

그림 4는 Lazy & Closure의 추론 방식을 보여준다. 그림 4와 같은 두 트리플 집합에서 변경부분을 구한다고 하자. (C subClassOf A) 트리플은 K, K' 두 모델에 존재하기 때문에 두 트리플에 추론을 적용할 필요가 없게 된다. (B subClassOf A)와 (B subClassOf C)는 구조적 변경(△E)으로 보면 삽입과 삭제에 해당되는 트리플들이다. 하지만 정리 1과 같이 두 트리플의 주어가 같기 때문에 추론을 통해 서로 대응될 수 있는 가능성을 가진 트리플들이기 때문에 각각 C(K), C(K') 를 계산해서 대응될 수 있는지 검사해야 한다. 반면에 K' 모델의 (D subClassOf A) 트리플의 경우 K 모델에 D를 주어로 갖고 술어가 subClassOf인트리플이 없기 때문에 추론을 계산 할 필요가 없다. 이와 같이 불필요한 C(K), C(K')에 대한 계산을 피할 수 있게 되므로 변경탐지에 효율적으로 사용할 수 있다. 이처럼 구조적 변경을 한

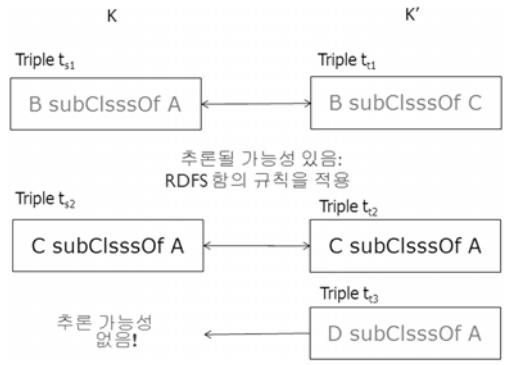


그림 4 Lazy & Closure 변경 탐지에서의 추론

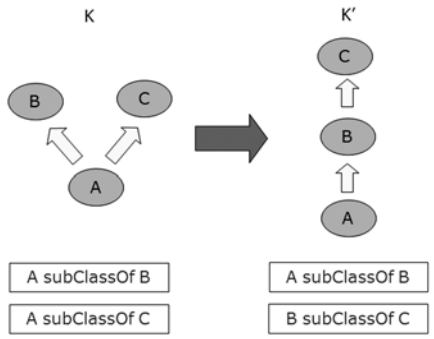


그림 5 트리플 차집합을 사용할 때의 예외

결과에서 주어가 같은 트리플들만을 추론하는 전략은 불필요한 추론을 줄여주는 효과를 가진다. 하지만 그림 5와 같은 경우를 생각해보자. 두 모델 K, K'의 구조적 변경을 구하면 차집합 계산 후 트리플(A subClassOf C)는 삭제 연산에 해당되며 K' 모델에서 A를 주어로 가지는 트리플이 없으므로 K'의 어떤 트리플과도 대응되지 않으므로 변경 내용에 포함된다. 하지만 (A subClassOf B) 트리플의 추이 관계를 통해 (A subClassOf C)의 트리플이 생성되므로 변경이라고 볼 수 없다. 이와 같은 경우를 위해 단순한 트리플의 차집합 연산을 사용할 수 없으며 확장이 필요하게 된다. 본 Lazy & Closure 기법에서는 트리플이 공통으로 두 모델에 있어도 주어가 같은 트리플이 2개 이상 있을 경우 구조적 변경에 해당하는 차집합 연산을 적용하지 않는다. 어차피 차집합 연산에 적용을 안 받더라고 추론 연산에서 계산되기 때문에 변경 내용에 차이는 없어지게 된다.

정리 2. 트리플의 술어에 해당되는 추론 규칙만이 적용된다.

변경 탐지에서의 추론의 이용에 있어 모든 추론 규칙을 적용할 필요는 없다. 그림 4의 예처럼 subClassOf 술어는 해당되는 규칙 11만 적용되면 된다. 술어가 type이라면 규칙 9가 적용되어야 하고 subPropertyOf 술어의 경우 규칙 5와 규칙 7이 적용된다.

그림 6은 이러한 특성을 고려한 Lazy & Closure 변경 탐지 알고리즘이다. 먼저 두 모델에서 변경되지 않는 공통 부분들을 Ex_Set 함수를 이용하여 삭제한다. Ex_Set(K, K')은 앞에서의 확장된 구조적 변경으로 두 모델 K, K'에서 명확하게 추론되지 않는 트리플들을 걸러주게 된다. 이렇게 걸러진 두 모델 사이에서 주어가 같은 트리플들에 대해서 ApplyRule 함수를 적용한다. ApplyRule(x)은 정리 2와 같이 x의 트리플에 포함된 술어에 해당되는 규칙들만을 적용하게 된다. 이 때 비교되는 트리플 쌍에 적용하는 선택된 RDFS의 함의 규칙의 순서는 [18]에서 사용한 최적화된 함의 규칙의 순서를 적용한다.

5. 실험 결과

본 실험은 리눅스 환경의 2GByte 메인 메모리를 가진 Pentium 4-3.2Hz에서 수행되었다. RDF 파서로 Rio 파서를 이용하였으며 RDFS 추론은 Sesame[19]의 추론 엔진을 사용하였고 JAVA로 구현되었다. 기존의 RDF 모델에 대한 변경 탐지 기법들은 3장에서 설명한 3개의 변경 탐지 기법의 형태로 분류할 수 있다. 따라서 본 실험에서는 3개의 변경 탐지 기법과 논문에서 제안한 Lazy & Closure 기법으로 2가지 실험을 통해 비교를 하였다. 우선 두 RDF 모델에서 변경 부분의 크기이다.

```
알고리즘 (Lazy & Closure)
01:  Input : S_source = Set of triples in source model
02:          S_target = Set of triples in target model
03:  Output : Set of change operation Diff using entailment rules
04:  DO {
05:       // 두 모델에 공통으로 나타나는 트리플을 확장된 구조적 변경으로 삭제한다.
06:       S_source = Ex_Set (S_source – S_target )
07:       S_target = Ex_Set (S_target – S_source)
08:    For every key in L_key  // L_key : 모델에 있는 남아 있는 트리플 주어들의 리스트
09:       Select all triples which satisfy the same subject in S_source
10:       Select all triples which satisfy the same subject in S_target
11:            For every possible triple pair (x, y), x∈ S_source , y∈ S_target,
12:                 x' = ApplyRule (x) // x'은 함의 규칙에 의해 x로부터 추론된 트리플
13:                    if (x' == y)
14:                    else x  ∪ Diff as deletion
15:                 y' = ApplyRule (y)  // y'은 함의 규칙에 의해 y로부터 추론된 트리플
16:                    if (y' == x)
17:                    else y  ∪ Diff as insertion
18:       // pair (x, y)에 해당되지 않는 트리플은 추론을 하지 않는다
19:            Add triples Diff as deletion or insertion
20:       } While (L_key  is not empty)
```

그림 6 Lazy & Closure 변경 탐지 알고리즘

표 2 Gene Ontology TermDB RDF 데이타

데이타 / 속성	G1	G2	G3	G4	G5	G6	G7	G8
트리플 수	397720	404892	409671	413923	415488	418684	418927	420036
추론된 트리플 수	599298	608336	614238	628964	631497	633409	634888	637292
날짜 (연도-월)	07-11	07-12	08-01	08-02	08-03	08-04	08-05	08-06
크기(MB)	31	31	32	32	32	32	33	33

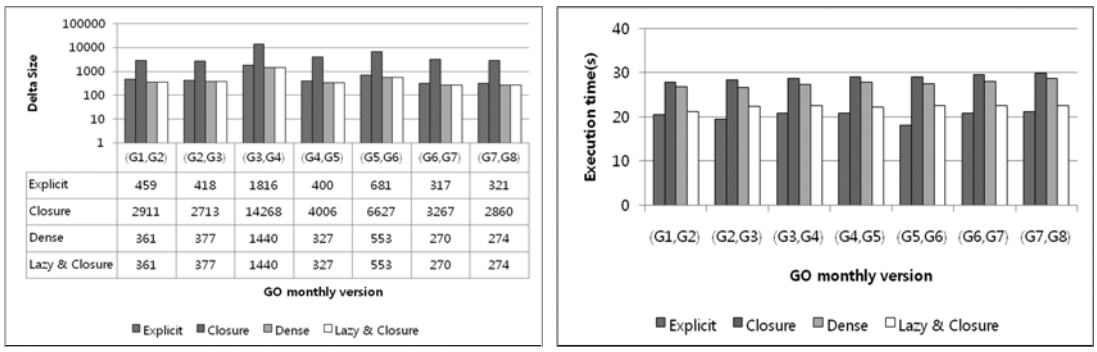
표 3 Uniprot Taxonomy RDF 데이타

데이타 / 속성	U1	U2	U3	U4	U5
트리플 수	2637046	2703674	2725324	2755810	2829621
추론된 트리플 수	8035785	8228086	8285704	8368233	8565134
날짜 (연도-월-일)	08-02-26	08-04-08	08-04-29	08-06-10	08-07-01
크기(MB)	187	192	193	195	201

변경 부분의 크기는 정확하면서도 크기에 있어 작을수록 적은 비용을 가지게 된다. 두 번째로는 변경 부분을 빠르게 찾아내는 실행 시간을 비교하였다. 본 논문에서 실험한 데이타는 Gene Ontology TermDB RDF 데이타와 Uniprot Taxonomy RDF 데이타이며 각각의 데이타 속성은 표와 같다. 두 데이타 모두 subClassOf 관계만을 가지고 있으며 인스턴스의 type이나 subPropertyOf 같은 술어는 가지고 있지 않다.

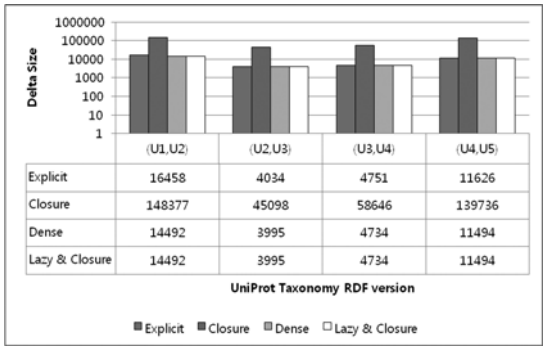
그림 7은 Gene Ontology 데이타에 대한 실험 결과를 보여주고 있다. 그림 7(a)의 가로축은 비교되는 데이타 쌍을 보여주며 세로축은 변경 탐지 결과를 보여준다. 변경 탐지 결과는 삽입과 삭제되는 트리플의 수를 의미하며 실험에서는 subClassOf의 트리플 수를 고려하였다. 결과에서 알 수 있듯이 Dense 기법과 Lazy & Closure 기법이 가장 작은 변경 탐지 결과를 보여주었으며 Closure의 경우 3.3절에서 언급하였듯이 오히려 변경 탐지 결과가 Explicit 기법보다 더 많은 것을 알 수 있다. 또한 그림 7(b)의 실행 시간을 보게 되면 Dense 기법의 경우 적은 변경 탐지 결과를 보여주는 대신에 실행 시간에 있어 비효율적인 반면 Lazy & Closure 기법은 Dense 기법보다 더 좋은 성능을 보여주고 있다. Dense 기법이 RDF 모델의 전체를 추론하는 반면에 Lazy & Closure 기법은 변경 탐지 결과에 포함될 수 있는 가능성을 가진 트리플을 추론하기 때문이다.

그림 8은 Uniprot 데이타에 대한 실험 결과를 보여주고 있다. 이전 실험과 마찬가지로 변경 탐지 결과에 있어서는 그림 8(a)에서 보듯이 Dense 기법과 Lazy & Closure 기법이 작은 변경 결과를 보인다. 하지만 그림 8(b)의 성능을 고려한 결과에서는 Explicit 기법과 Lazy & Closure 기법이 우수함을 알 수 있다. 따라서 두 가

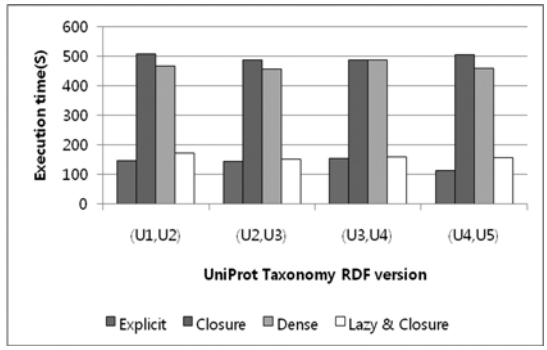


(a) 변경 탐지 결과(subClassOf 트리플의 수) (b) 변경 탐지 실행 시간

그림 7 Gene Ontology에 대한 실험 결과



(a) 변경 탐지 결과(subClassOf 트리플의 수)



(b) 변경 탐지 실행 시간

그림 8 Uniprot 데이타에 대한 실험 결과

지 결과를 종합해 보면 Lazy & Closure 기법이 좀 더 유연함을 알 수 있다. 또한 앞의 Gene Ontology 실험과 비교를 해보면 대용량일수록 Lazy & Closure 기법이 다른 기법보다 더 작은 변경 내용과 효율성을 보인다고 볼 수 있다. 이와 같은 실험 결과는 Lazy & Closure은 줄일 수 있는 변경 부분에만 추론을 적용하는 후방향 추론 기법을 사용하기 때문이다.

6. 결론 및 향후 연구 계획

최근의 RDF 모델의 변경 탐지 기법에 대한 연구들은 기존의 구조적 변경 내용뿐만이 아니라 RDFS의 함의 규칙을 이용하여 좀 더 정확하면서도 적은 변경 부분을 찾으려는 연구들이 주를 이룬다. 하지만 이런 변경 탐지는 전방향 추론을 한 후에 두 모델에서 변경 부분을 찾고 있으며 이런 접근 방식은 비교되는 모델의 크기와 비교 시간의 증가라는 문제점이 생기며 불필요한 변경 부분을 만들기도 한다.

본 논문에서는 이러한 문제점을 해결하기 위해 변경 부분에 대해서만 RDFS 함의 규칙을 적용하는 효율적인 변경 탐지 기법을 제안하였다. 제안하는 기법은 우선 두 모델에 대해 확장된 구조적 변경을 취해 추론에 적용 받지 않는 트리플들을 걸러준 후에 후방향 추론과 트리플에 적용되는 규칙들만을 적용을 하였다. 이러한 접근 방법은 기존의 전방향 추론을 이용하는 변경 탐지 기법과 같은 정확도를 보이며 변경 탐지 시간에 있어 우수한 성능을 보인다.

향후 연구로서 본 논문에서는 RDFS 추론을 기반으로 하지만 OWL과 같은 온톨로지에서는 더 다양하고 복잡한 연구가 필요하다. 또한 대용량의 온톨로지의 경우 데이타베이스 기반의 추론을 이용하는 변경 탐지 기법에 대한 연구가 필요하다.

참 고 문 헌

[1] Graham Klyne, Jeremy J. Carroll and Brian McBride, "Resource Description Framwork(RDF): Concepts and Abtrract Syntax," W3C Recommendation, 2004.

[2] Giorgos Flouris, Dimitris Manakanatas, Haridimos Kondylakis, Dimitris Plexousakis and Grigoris Antoniou, "Ontology Change: classification and survey," The Knowledge Engineering Review, 23(2), 2008.

[3] Michel Klein, Atanas Kiryakov, Damyan Ognyanov and Dieter Fensel, "Ontology Versioning and Change Detection on the Web," In Proceedings of EKAW, 2002.

[4] Ora Lassila, Ralph R. Swick, eds, "Resource Description Framework (RDF) Model and Syntax Specification," http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/.

[5] Max Volkel and Tudor Groza, "SemVersion: An RDF-Based Ontology Versioning System," In Proceedings of ICWI, 2006.

[6] Dimitris Zeginis, Yannis Tzitzikas and Vassilis Christophides, "On the Foundation of Computing Deltas between RDF Models," In Proceedings of ISWC, 2007.

[7] Jeremy J. Carroll, "Signing RDF Graphs," In Proceddings of ISWC, 2003.

[8] Tim Berners-Lee and Dan Connolly, "Delta: An Ontology for the Distribution of Differences Between RDF Graphs," http://www.w3.org/DesignIssues/Diff, 2004.

[9] 이동희, 임동혁, 김형주, "내포된 공노드를 포함하는 RDF 문서의 변경 탐지 기법," 정보과학회논문지:데이타베이스, 34(6), 2007.

[10] Natalya F. Noy and Mark A. Musen, "PromptDiff: A Fixed-Point Algorithm for Comparing Ontology Versions," In Proceedings of AAAI, 2002.

[11] Johann Eder and Karl Wiggisser, "Change Detec-

tion in Ontologies Using DAG Comparison," In Proceedings of CAiSE, 2007.
[12] Jeen Broekstra and Arjohn Kampman, "Inferencing and Truth Maintenance in RDF Schema," In Proceedings of the Workshop on Practical and Scalable Semantic System, 2003.
[13] Charles Forgy, "Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem," Artificial Intelligence, 19(1), 1982.
[14] Heiner Stuckenschmidt and Jeen Broekstra, "Time-Space Trade-offs in Scaling up RDF Schema Reasoning," In Proceedings of the workshop on Scalable Semantic Web Knowledge Base System (WISE 2005), 2005.
[15] Yuanbo Guo, Zhengxiang Pan and Jeff Heflin, "An Evaluation of Knowledge base Systems for Large OWL Datasets," In Proceedings of ISWC, 2004.
[16] Patric Hayes and Brian McBride, "RDF Semantics," Technical Report, W3C Recommendation, 2004.
[17] V. Christophides, D. Plexousakis, M. Scholl and S. Tourtounis, "On Labeling Schemes for the Semantic Web," In Proceedings of WWW, 2003.
[18] 김기성, 유상원, 이태휘, 김형주, "RDF 스키마 함의 규칙 적용 순서를 이용한 RDFS 추론 엔진의 최적화", 정보과학회논문지:데이타베이스, 33(2), 2006.
[19] Jeen Broekstra, Arjohn Kampman and Frank van Harmelen, "Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema," In Proceedings of ISWC, 2002.

임 동 혁
2003년 고려대학교 컴퓨터교육과(학사) 2005년 서울대학교 컴퓨터공학부(석사) 2005년~현재 서울대학교 컴퓨터공학부 박사과정 재학 중. 관심분야는 데이타베이스, 시맨틱 웹, 온톨로지

김 형 주
1982년 서울대학교 전산학과(학사). 1985년 Univ. of Texas at Austin(석사) 1988년 Univ. of Texas at Austin(박사). 1988년 9월~1990년 12월 Georgia Institute of Technology(부교수). 1991년~현재 서울대학교 컴퓨터공학부 교수 관심분야는 데이타베이스, XML, 시맨틱 웹, 온톨로지