메타 규칙과 번역의 혼용을 통한 규칙엔진 기반 OWL 추론 엔진의 성능 향상 방법

장민수O, 손주찬, 조영조
한국전자통신연구원
{minsuO, jcsohn, youngjo}@etri.re.kr

Efficient Rule-based OWL Reasoning by Combining Meta Rules and Translation

Minsu Jang, Joo-Chan Sohn, Youngjo Cho

요 약

생성 규칙(Production Rule)과 이를 기반으로 하는 규칙 엔진(Rule Engine)을 기반으로 한 OWL 추론 엔진은 메타 규칙(Meta Rule)에 의존해 왔다. 메타 규칙은 OWL의 의미론(Semantics)을 표현하기 용이하여 보다 손쉽게 OWL 추론 엔진을 구현할 수 있다는 장점을 제공하였으나 OWL 추론 성능에 있어 추론 속도와 대용량 온톨로지 처리 측면에서 모두 만족할 만 한 성과를 얻지 못하였다. 본 논문은 DLP(Description Logic Programming)의 번역 접근법을 기반으로 한 번역 규칙(Translation Rules)을 메타 규칙과 혼용하는 OWL 추론 기법을 소개한다. LUBM 벤치마크를 통해 이 기법이 메타 규칙만을 이용했을 때 보다 100% 이상 추론 성능을 향상시켰을 뿐 아니라 메모리 사용량도 대폭 축소시켰음을 확인할 수 있었다. 또한, 번역을 통해 제한없는 차수 제약(Cardinality Restriction) 관련 추론을 지원하는 등 보다 넓은 범위의 OWL 추론을 지원할 수 있다.

1. 서 론

Description Logic 외의 논리 시스템을 기반으로 OWL 추론 엔진을 구축하려는 다양한 시도가 있다. 대표적인 예로 일차논리(First-Order Logic) 증명기를 기반으로 한 Hoolet[1], 논리 프로그래밍의 파생 시스템들을 기반으로 한 F-OWL[2], Euler[3], KAON2[4], 생성 규칙(Production Rule)을 기반으로 한 OWLJessKB[5], Jena[6], Bossam[7] 등을 들 수 있다. Description Logic 외의 논리 시스템을 기반으로 OWL 추론 엔진을 구축하려는 시도를 하게 되는 이유는 다음과 같이 요약해 볼 수 있다.

- 잘 연구된 논리 시스템과 구현물들을 재활용할 수 있다.
- Description Logic 기반 추론 엔진의 단점을 보완할 수 있다. 대용량 ABox 추론[4], 노미널(Nominal) 추론 등은 Description Logic 기반 OWL 추론 엔진들이 잘 처리하지 못한다.
- OWL의 범위를 넘어서는 표현력을 활용할 수 있다. Datalog나 생성 규칙을 이용하면 NAF(Negation As Failure), 속성 결합(Property Composition)[8] 등을 이용하여 OWL로 표현할 수 없는 지식을 표현할 수 있다.

생성 규칙 및 이를 기반으로 한 규칙 엔진(Rule Engine)은 실용 분야에 광범위하게 활용되고 있는 논리 기반 시스템이다. 규칙 엔진에 OWL 추론 기능을 부여하여 확장함으로써 규칙 기반 응용의 웹 확장을 용이하게 도모할 수 있을 뿐 아니라 OWL 추론을 실세계 응용에서 활용할 수 있는 가능성을 높일 수 있다. 우리는 본 논문을 통해 규칙 엔진을 기반으로 한 OWL 추론 시스템의 구축에 있어 추론 성능을 향상시킬 수 있는 기법을 제안하고자 한다.

대표적인 규칙 기반 OWL 추론 엔진으로는 OWLJessKB, Jena, Bossam 등이 있다. 규칙 기반 OWL 추론 엔진들은 몇가지 문제점을 노출하고 있는데 그 중 대표적인 것이 불완전한 추론 및 낮은 성능이다.

OWL이 기반으로 하고 있는 Description Logic과 생성 규칙 간에는 상호 호환되지 않는 표현력 영역이 존재한다. 이 때문에 규칙을 기반으로 한 OWL 추론 엔진은 추론의 완전성(Completeness)을 추구할 수 없다[8,9]. 따라서, 규칙 기반 OWL 추론 엔진의 입장에서는 OWL 추론 영역 중 지원할 수 있는 영역의 크기가 어느 정도인가가 관심사가 된다. 상기한 규칙 엔진 기반 OWL 추론기들은 OWL 의미론을 기술한 메타 규칙(Meta Rule)을 이용하여 추론하는데 이 방식은 임의 개수의 클래스를 포함하는 불린 결합(Boolean Composition)이나 차수 제약(Cardinality Restriction)을 표현하는데 한계를 드러낸다. 이는 OWL 추론 영역을 제한하는 대표적인 원인으로 작용한다.

규칙 기반 OWL 추론 엔진의 낮은 OWL 추론 성능에 대하여는 LUBM 벤치마크를 통해 보고된 바 있다[10]. 우리는 이러한 낮은 성능의 원인이 메타 규칙의 활용에

있다고 판단하였다. 메타 규칙을 활용한 접근 방법에서는 OWL 문서를 구성하는 RDF 트리플(Triple)들을 일련의 사실(Fact)로 번역한 후 메타 규칙을 적용하여 추론하는 방식을 취하므로 사실의 개수가 많아진다. 또한, 메타 규칙은 많은 경우 트리플로부터 생성된 사실의 모든 항(Term)에 대해 패턴 매칭을 시도하게 된다. 사실 개수의 증가와 매칭 대상 항 개수 증가는 규칙 엔진의 주요한 성능 저하 원인이 된다.

우리는 상기한 바와 같은 메타 규칙 기반의 OWL 추론이 갖는 문제점을 개선하기 위해 메타 규칙과 더불어 번역(Translation)을 활용하는 혼용법을 시도하였다. 번역은 OWL 문서를 일방적으로 사실로 번역하지 않고 규칙으로도 번역하는 접근법을 말한다. 번역을 통해 사실의 개수뿐 아니라 패턴 매칭 대상 항(Term)의 개수도 줄일 수 있다. 우리는 혼용법을 우리 엔진(이하 BOSSAM이라 명함)에 구현한 후 LUBM으로 벤치마크하여 메타 추론만을 활용했을 때보다 두 배 이상 성능이 향상됨을 확인할 수 있었다.

이하 논문의 구성은 다음과 같다. 2장에서 혼용법을 보다 상세히 설명한다. 3장에서 벤치마크 결과를 제시 및 분석한다. 그리고, 4장에서 결론 맺는다.

2. 메타 규칙과 번역의 결합

OWL Lite/DL과 규칙은 서로 다른 논리 시스템을 기반으로 한다. 전자는 Description Logic을, 후자는 혼논리(Horn Logic)를 기반으로 한다. 따라서, 규칙을 이용하여 OWL 추론을 수행하려면 서로 다른 두 논리 시스템을 연결해 주어야 한다. 연결 방법은 크게 두 가지로서 메타 추론(Meta Reasoning)과 번역(Translation)이다.

추론 대상 논리 시스템을 L1이라 하고 추론 수행 논리 시스템을 L2라 하자. 규칙을 이용한 OWL 추론의 경우 OWL이 L1이 되고 규칙이 L2 가 된다.

메타 추론 접근법은 L1 지식베이스를 대상으로 한 추론을 수행하기 위해 L1의 의미론을 L2를 이용하여 추론 규칙으로 기술한다. 본 논문에서는 이 추론 규칙을 메타 규칙으로 부른다. 즉, 메타 규칙 접근법은 L2 기반 위에 L1 추론 기계를 구축한다.

번역 접근법은 L1의 문장을 L2의 문장으로 번역하는 사상(Mapping) 또는 번역 규칙(Translation Rule)을 이용한다. L1 지식베이스는 번역 규칙에 의거하여 L2 지식베이스로 번역되고, 번역된 후에는 L2 추론 시스템을 통해 추론된다.

2.1. 메타 규칙 기반 OWL 추론

OWL 의미론[13]은 증명이론적(Proof-Theoretic) 메타 규칙으로 공리화(Axiomatization)된다. OWL 의미론과 규칙 언어의 표현력 및 해석 모델 차이로 인해 OWL 의미론을 완전하게 메타 규칙으로 공리화하는 것은 불가능하지만 [1,8,9], 대부분의 주요한 OWL 의미론은 용이하게 메타 규칙으로 적을 수 있다. 이는 Euler[3], OWLJessKB[5], Jena[6], Bossam[7] 등의 성공적인 규칙 기반 OWL 추론 엔진들을 통해 확인할 수 있다.

다음은 OWL 추론을 위한 메타 규칙의 예이다. 상기한 OWL 추론 엔진들이 공히 이런 규칙을 활용한다.

R1: rdfs:subClassOf(?c,?d) ∧ rdfs:subClassOf(?d,?e)
→ rdfs:subClassOf(?c,?e)

R2: rdfs:subClassOf(?c,?d) ∧ ?c(?i) → ?d(?i)

R1은 클래스 간 상하위 관계의 이행성(Transitivity)을 기술하고 있고, R2는 상하위 관계에 있는 클래스의 부분 집합 관계를 기술한다.

대부분의 OWL 의미론은 이와 같이 메타 규칙으로 용이하게 표현 가능하나, 표현하기 어렵거나 불가능한 것들도 있다. 임의 개수의 클래스 간에 맺어지는 불린 조합(Boolean Composition)과 차수 제약(Cardinality Restriction)을 포함하는 클래스 기술(Class Description)은 대표적인 예이다. 예를 들어, 두 클래스의 접합(Intersection)으로 정의된 클래스에 속한 개체(Individual)의 타입을 유추하는 메타 규칙은 다음과 같이 적을 수 있다.

R3: owl:intersectionOf(?c,?l)
∧ rdf:first(?l,?d) ∧ rdf:rest(?l,?r1)
∧ rdf:first(?r1,?e) ∧ rdf:rest(?r1,rdf:nil)
∧ ?d(?i) ∧ ?e(?i)
→ ?c(?i)

R3의 조건부에서 보듯이 R3는 클래스 두 개로 구성된 접합만을 처리할 수 있다. 임의 개수의 클래스가 개입된 접합을 처리할 수 있는 메타 규칙을 기술하기는 매우 어렵다. 상기한 owl:intersectionOf 뿐 아니라 owl:unionOf나 owl:oneOf와 같이 목록 구조를 포함하는 OWL 구문들도 마찬가지로 메타 규칙을 적기 어렵다. RDF 목록의 자기 복제 구조(Recursive Structure)를 기반으로 목록 구조를 처리할 수 있는 메타 규칙을 기술할 수도 있다. 그러나, 이러한 프로그램은 중간 결과로 많은 수의 사실(Fact)을 생성하게 되므로 과도한 계산 부하를 발생시켜 OWL 추론 성능을 저하시킨다.

R3에서 보인 바와 같은 난점이 차수 제약 처리에도 존재한다. 아래의 R4 규칙은 최대 차수(Maximum Cardinality) 제약을 기반으로 상충(Inconsistency)을 찾아낸다. 이 규칙은 최대 차수 값이 2인 경우만 처리할 수 있다. 차수 값에 따라 조건부의 owl:differentFrom 패턴의 개수가 달라져야 함을 발견할 수 있다. 즉, R3와 마찬가지로 구조적으로 차수 N에

대해 일반화된 메타 규칙을 기술하는 것은 불가능하다.

R4: owl:equivalentClass(?c,?r)
∧ owl:Restriction(?r) ∧ owl:onProperty(?r,?p)
∧ owl:maxCardinality(?r,2)
∧ ?p(?x,?y1) ∧ ?p(?x,?y2) ∧ ?p(?x,?y3)
∧ owl:differentFrom(?y1,?y2)
∧ owl:differentFrom(?y2,?y3)
∧ owl:differentFrom(?y1,?y3)
→ Inconsistency

성능 측면을 살펴본다. R2, R3를 보면 술어 부호(Predicate Symbol)에 변수가 사용되고 있음을 볼 수 있다. 이와 같은 2차 술어(Second-Order Predicate) 구문은 패턴 매칭에 큰 부담을 안긴다. 예를 들어, R2의 조건부에 쓰인 패턴 ?c(?i)는 모든 일항사실(1-ary Facts)들에 매칭되고, R4의 조건부에 쓰인 ?p(?x,?y1)은 모든 2항사실(2-ary Facts), 즉 모든 트리플에 매칭된다. 이는 곧바로 논리곱(Conjunction)에 의한 결합(Join)에 연쇄적으로 매우 큰 부하로 작용하게 되는데, 이로 인한 추론 성능의 저하는 심각해진다. 술어 부호가 상수라면 각 패턴에 대한 매칭 부하가 큰 폭으로 줄어들 뿐 아니라 결합 부하는 더 큰 폭으로 줄어들게 된다. 메타 규칙을 이용한 개체 관련 추론, 즉 ABox 추론은 상기한 바와 같은 2차 술어 구문을 반드시 포함하게 된다. 따라서, 메타 규칙은 추론 성능을 크게 저하시킨다.

2.2. 번역 기반 OWL 추론

번역 기반 OWL 추론에서는 OWL 문서를 구성하는 문장들을 규칙으로 번역한다. 번역 과정을 거치면 OWL 문서는 규칙베이스로 변환된다. 규칙 엔진을 이용하여 규칙베이스에 대해 추론을 수행하면 자연스럽게 OWL 추론을 수행하게 된다.

Grosof 등은 OWL DL로부터 혼규칙(Horn Rule)으로 번역하는 번역 규칙을 제안한 바 있다[8]. 규칙은 혼논리의 파생 시스템이므로 Grosof 등의 번역 규칙을 적용할 수 있다. 다음은 [8]에 소개된 번역 규칙에 의거한 번역의 예이다.

M1: (C rdfs:subClassOf D) → {C(?i) → D(?i)}
M2: (C owl:intersectionOf (D E))
→
{D(?i) ∧ E(?i) → C(?i)}

상기의 예에서 보듯이 번역은 개별 OWL 문장에 대해 적용된다. 메타 규칙이 OWL 의미론을 일반화된 규칙으로 기술하는 것과 다름을 알 수 있다. 구체적으로 살펴보면, 번역의 결과로 생성되는 문장들은 메타 규칙의 인스턴스(Instance)에 해당됨을 볼 수 있다. 즉, 번역의 결과로 생성되는 문장들은 메타 문장보다 한 단계 아래의 추상화 수준에 있다고 볼 수 있다. 번역의 결과로 생성되는 문장들을 번역 문장(Translated Sentences)이라고 할 때, 메타 규칙과 개념적인 관계는 그림 1과 같이 표현 가능하다.

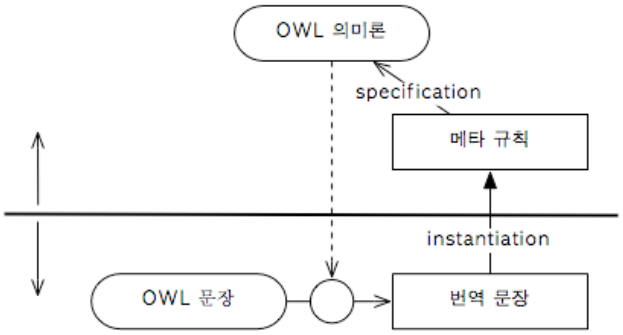

그림 1. 메타 규칙과 번역 문장의 관계

번역 과정은 번역 규칙에 의거하여 자동으로 수행된다. 자동화된 번역 메카니즘을 통해 RDF 목록 구조 및 차수 제약으로부터 번역 문장을 용이하게 생성해낼 수 있다. 따라서, 2.1에서 언급한 목록 구조를 포함하는 OWL 구문 및 차수 제약을 손실없이 처리할 수 있다. 메타 규칙보다 더 넓은 영역의 OWL 의미론을 처리할 수 있는 셈이다.

성능 측면에서 보면, 앞서 언급한 대로 번역 문장은 메타 규칙의 인스턴스로 볼 수 있고 따라서 2차 술어 구문을 포함하지 않으므로 규칙 엔진의 패턴 매칭 부담이 대폭 줄어든다. 따라서 번역은 메타 규칙에 비해 높은 추론 성능을 보장한다.

한가지 번역의 문제점은 어휘 추론, 즉 TBox 추론에 있다. OWL의 의미론은 외연적 의미(Extensional Meaning)를 기반으로 하고 있으며 번역 규칙도 이를 따르므로 번역 문장은 개체(Individual)에 대한 추론만 표현한다. 예를 들어, M1과 같이 번역 문장을 통해 클래스 간의 관계를 기반으로 개체의 타입을 결정할 수는 있으나, 2.1에 보인 R1과 같이 클래스 간의 관계 자체를 유도할 수 없다. 일부 번역 기반 OWL 추론 엔진은 귀납추론(Induction)을 통해 간접적으로 TBox 추론을 시도하기도 한다. A가 B의 하위 클래스인지 판단하기 위해 A의 개체를 임의로 생성한 후 추론을 수행하여 그 개체가 B의 개체이기도 한 지 여부를 보는 방식이다[15].

2.3. 번역과 메타 규칙의 결합

1장에서 언급한 바와 같이 규칙 엔진을 기반으로 한 OWL 추론 엔진들은 메타 규칙을 기반으로 한다. 우리는 번역 기법을 메타 규칙과 혼용함으로써 추론 성능 및 완전성을 향상시키고자 하였다. 기본 아이디어는 번역 기법을 통해 ABox 추론을 수행하고 메타 규칙을 통해 TBox 추론을 수행한다는 것이다. 번역은 추론 속도를 대폭 향상시키는데 기여하고 메타

규칙은 TBox 추론을 가능하게 한다. TBox 추론 영역만 보면 2차 술어 구문이 사용되지 않으므로 전체 성능에 큰 영향을 주지 않는다.

번역과 메타 규칙을 결합한 OWL 추론 시스템의 얼개를 그림으로 나타내 보면 그림 2와 같다.

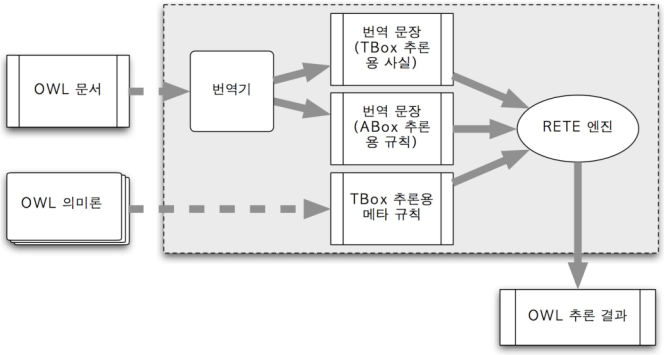

그림 2. 메타 규칙과 번역 결합형 OWL 추론 시스템의 얼개

번역기는 입력된 OWL 문서를 번역 문장으로 번역한다. 번역 문장은 규칙과 사실로 구성된다. 규칙은 ABox 추론에 활용되고, 사실은 메타 규칙과 결합되어 TBox 추론을 수행하는데 활용된다.

메타 규칙과 번역을 결합한 OWL 추론 절차는 표 1에 보는 바와 같다. S는 입력 OWL 문장의 집합, Mr은 메타 규칙, TRr은 번역 규칙, Tr과 Tf는 각각 번역의 결과로 생성되는 번역된 규칙 및 사실을 나타낸다.

표 1. 메타 규칙과 번역 결합형 OWL 추론 절차

```
CombinedReasoning(S, Mr, TRr)
{
  1) TRr을 기반으로 S를 번역: S → {Tr, Tf}
  2) KB = Mr ∪ Tr ∪ Tf
  3) KB에 대해 추론 수행: KB ⊢ S
  4) S == {∅}이면 종료, 아니면 1)로 이동.
}
```

3. 성능 시험 결과

우리는 메타 규칙 기반의 OWL 추론 엔진인 BOSSAM에 번역 접근법을 추가함으로써 본 논문에서 제시한 바와 같은 메타 규칙 및 번역 혼합형 엔진을 개발하였다. 성능 향상 여부 판단을 위해 기존의 메타 규칙 기반 엔진(이하 MetaBOSSAM으로 부르기로 함)과 혼합형 엔진(HybeBOSSAM으로 부르기로 함)의 성능을 상호 비교하였다.

MetaBOSSAM은 96개의 OWL DL 추론을 위한 메타 규칙을 지니며 OWL DL의 모든 어휘를 지원한다. 재귀적인 처리를 통해 임의 개수의 클래스가 개입된 불린 결합을 지원하나, 차수 제약은 차수 값의 범위가 1 ~ 3인 경우에만 지원한다.

HybeBOSSAM은 49개의 메타 규칙을 지니며, MetaBOSSAM에 없는 번역기(그림 2 참조)를 포함한다. 번역을 통해 임의 개수의 클래스가 개입된 불린 결합을 지원하고, 차수 제약은 차수 값에 상관없이 지원한다. 또한, HybeBOSSAM은 집합의 포함 관계를 기반으로 한 클래스 간 상하위 관계를 처리할 수 있다. 예를 들어, L1, L2가 각각 클래스 기술(Class Description)의 집합이라 하자. A가 L1에 속한 클래스들의 접합(Intersection)으로 정의되고, B가 L2에 속한 클래스들의 접합으로 정의되었다고 할 때, L2가 L1의 부분집합이면 B는 A의 하위 클래스가 된다. 만약 L1과 L2가 동치이면 A와 B도 동치이다. OWL에 있어 이와 같은 집합 간 동치성 및 포함관계 유추는 매우 복잡한 계산을 필요로 한다. OWL은 UNA(Unique Name Assumption)을 기반으로 하지 않기 때문이다. 간단히 아이디 비교를 통해 두 클래스 또는 개체 간의 동치성을 판단할 수 없다. 따라서, 메타 규칙을 이용하면 두 집합 간의 동치성 및 포함 관계 판단을 위한 복잡한 프로그램을 구축해야 한다. 이는 앞서 언급한 바 대로 추론 성능을 크게 저하시킨다. 번역을 이용하면 OWL 문서에 기술된 각 집합마다 간단한 구조의 포함관계 및 동치성 판단 규칙을 생성하여 처리할 수 있다.

성능 시험을 위한 장비로 인텔 코어 규오 1.8Ghz CPU와 2GB 램을 장착한 노트북을 사용하였다. 추론 엔진은 자바로 구현되었으며, 실험을 위해 Java 2 Standard Edition 5.0 런타임을 활용하였다. 실험 데이터는 LUBM[10]을 사용하였다.

MetaBOSSAM과 HybeBOSSAM은 모두 RETE 기반의 전향 추론 엔진이므로, 성능 측정은 두 부문에 대해 이루어졌다. 첫째는 OWL 온톨로지를 로딩하여 전향 추론을 수행 완료할 때 까지 걸리는 로딩 시간이고, 둘째는 질의(Query)에 응답하는데 걸리는 질의 응답 시간이다.

표 2는 MetaBOSSAM과 HybeBOSSAM의 로딩 시간을 측정한 결과이다. 엔진 이름 아래 숫자는 자바 런타임에 부여한 히프 메모리(Heap Memory) 크기이다. 시간 단위는 밀리초(ms)이다.

트리플 개수	15499	28152	34909	54771	100912
MetaBOSSAM (256M)	5156	12625	실패	실패	실패
MetaBOSSAM (512M)	6282	12640	14499	27734	실패
MetaBOSSAM (1024M)	5266	10376	18673	26562	57875
HybeBOSSAM (256)	3718	6531	8266	13921	23187
HybeBOSSAM	3734	6468	8220	13937	23562

(512)					

표에서 보듯이 온톨로지의 규모가 커질 수록 혼합형 엔진인 HybeBOSSAM이 메타 규칙만을 사용하는 MetaBOSSAM에 비해 100% 정도의 성능 우위를 보이고 있음을 볼 수 있다. 또한, 메모리 사용량에 있어서도 HybeBOSSAM은 MetaBOSSAM에 비해 효율적임이 드러났다. 10만 트리플을 규모의 온톨로지에 대해 512MB를 사용한 MetaBOSSAM은 메모리 초과 에러로 인해 로딩이 실패하였다. 반면 HybeBOSSAM은 23초만에 로딩을 완료하였다. 추가적으로 HybeBOSSAM의 경우 512MB를 사용한 경우 23만 트리플 규모의 온톨로지를 메모리에 읽어 전향 추론을 마치는데 110초가 걸렸다.

그림 3과 그림 4는 각각 MetaBOSSAM과 HybeBOSSAM을 이용한 질의 응답 시간 기록을 보여준다. Q1 ~ Q14는 LUBM에서 제공하는 시험용 질의문이다[10]. 그림에 표시한 대로 MetaBOSSAM의 경우 메모리 사용량이 많은 관계로 자바 런타임의 메모리 크기를 1024MB로 잡았고, HybeBOSSAM의 경우엔 512MB로 잡았다. 실험 결과를 보면, 10만 트리플 규모의 온톨로지에 대한 질의 처리의 경우 모든 질의가 0.5초 이내에 처리되고 있음을 볼 수 있으며, 트리플 규모의 증가에 따라 완만하게 성능 저하가 일어나고 있음을 볼 수 있다. 단, MetaBOSSAM의 경우 23만 트리플에 대한 질의를 처리하지 못한 반면, HybeBOSSAM은 11개의 질의에 대해 0.5초 이내에 처리하고 있음을 볼 수 있다. 질의문 Q4, Q8, Q9에 대해서는 다소 급격한 성능 저하를 보여 3.5초 ~ 4초 내외의 성능을 보였다.

질의 응답 내용의 정확성 검증 결과 MetaBOSSAM과 HybeBOSSAM 양자 모두 정확성(Soundness)과 완전성(Completeness)을 만족시켰다.

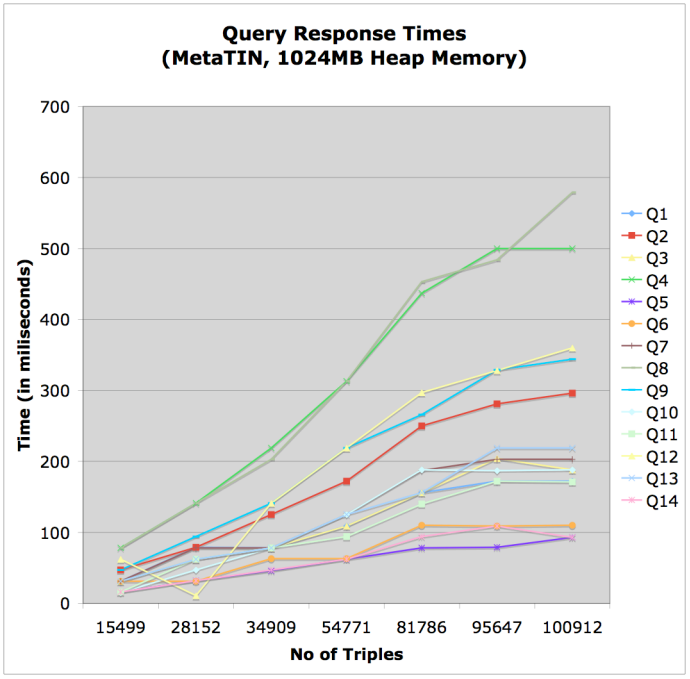

그림 3. MetaBOSSAM의 LUBM 질의 응답 시간

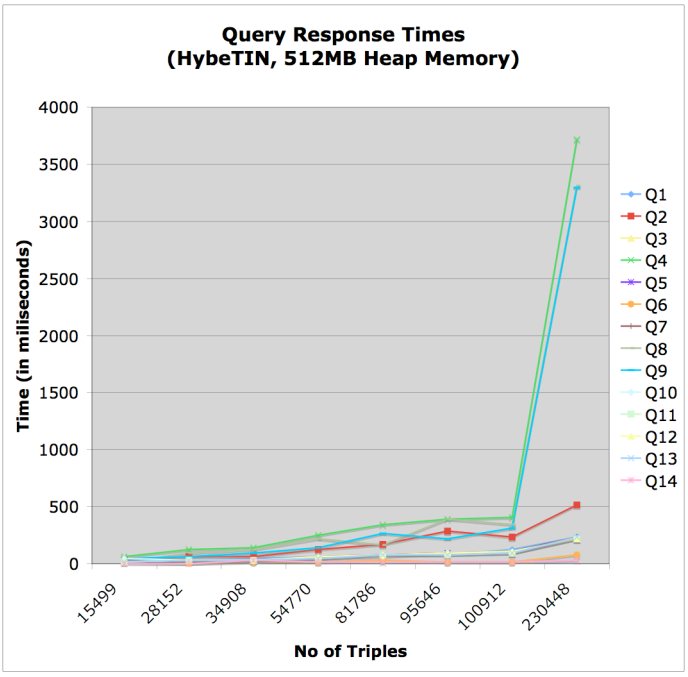

그림 4. HybeBOSSAM의 LUBM 질의 응답 시간

그림에서 보듯이 질의 응답 시간은 양 추론 엔진이 공히 훌륭한 성능을 보여주었으며 큰 차이를 보이지 않았다. 이는 BOSSAM이 전향 추론 엔진이라는 특성에서 기인하는 현상으로 보인다. 즉, 실제 추론이 수행되는 로딩 시간에 있어서는 MetaBOSSAM과 HybeBOSSAM이 큰 성능 차이를 보이지만, 추론을 수행하지 않는 질의 처리에 있어서는 큰 차이를 보이지 않는다는 점이다.

4. 결론

규칙 엔진은 실세계 응용을 위한 다양한 실용 기능을 제공한다. 규칙 엔진의 기능을 확장하여 OWL 추론을 지원함으로써 시맨틱웹 기술을 더 많은 실세계 응용에 활용할 수 있게 될 것으로 기대되며, 이러한 측면에서 규칙 엔진을 OWL 추론에 활용하는 것은 긍정적이다. 그러나, 그간의 규칙 엔진 기반 OWL 추론 엔진들은 낮은 성능을 보여 활용성에 있어 의문이 제기되곤 했다. 기존의 규칙 엔진 기반 OWL 추론 엔진들은 메타 규칙을 활용하여 OWL 추론을 수행하는데, 메타 규칙은 패턴 매칭 부하를 높일 뿐 아니라 OWL 의미론 처리 능력에 있어서도 한계를 보인다. 우리는 번역 기법을 적극 활용함으로써 메타 규칙 기반 추론 엔진의 한계를 극복하고자 하였다. 메타 규칙과 번역을 혼용하는 방법을 활용하여 실험한 결과 메타 규칙만 사용하였을 때 보다 두 배 이상 높은 추론 성능을 달성할 수 있었다. 더불어 새로운 접근 방법을 통해 임의 개수의 클래스를 포함하는 불린 결합 및 차수 제약 등 메타 규칙으로 처리하기 어려운 OWL 의미론 영역을 용이하게 처리할 수 있게 되었으며, 메모리 사용량도 대폭 줄일 수 있었다. 앞으로 추가적인 최적화를 통해 규칙 엔진 기반 OWL 추론 엔진을 다양한 실세계 응용에 적용할 수 있도록 만들고자 한다.

참고 문헌

1. Dmitry Tsarkov, Alexandre Riazanov, Sean Bechhofer, and Ian Horrocks: Using Vampire to reason with OWL. In Sheila A. McIlraith, Dimitris Plexousakis, and Frank van Harmelen, editors, Proc. of the 2004 International Semantic Web Conference (ISWC 2004), number 3298 in Lecture Notes in Computer Science. 471-485, 2004
2. Youyong Zou, Tim Finin and Harry Chen: F-OWL: an Inference Engine for the Semantic Web. In Proceedings of the third NASA-Goddard/IEEE Workshop on Formal Approaches to Agent-Based Systems. (2004)
3. Jos De Roo: Euler Proof Mechanism. At http://www.agfa.com/w3c/euler/. (2003)
4. U. Hustadt, B. Motik, U. Sattler: Reducing SHIQ Descrption Logic to Disjunctive Datalog Programs. In Proc. of the 9th International Conference on Knowledge Representation and Reasoning (KR2004), (2004) 152-162
5. Kopena, J. B., and Regli, W. C.: DAMLJessKB: A tool for reasoning with the semantic web. In Proc. of the 2003 International Semantic Web Conference. (2003)
6. Jeremy J. Carroll, Ian Dickinson, Chris Dollin, Dave Reynolds, Andy Seaborne, and Kevin Wilkinson: Jena: ImplemenBossamg the Semantic Web Recommendations. In Proceedings of the 13th International World Wide Web Conference. (2004)
7. Minsu Jang, Joo-chan Sohn: Bossam: an extended rule engine for the web. In Proceedings of RuleML 2004 (LNCS Vol. 3323), (2004)
8. Benjamin N. Grosof, Ian Horrocks, Raphael Volz, and Stefan Decker: Description logic programs: combining logic programs with description logic. In Proceedings of the twelfth international conference on World Wide Web. (2003)
9. T. Eiter, G. Ianni, and A. Polleres: Reasoning with Rules and Ontologies. (2006)
10. Y. Guo, Z. Pan, and J. Heflin: An Evaluation of Knowledge Base Systems for Large OWL Datasets. In Proceedings of Third International Semantic Web Conference, Hiroshima, Japan, LNCS 3298, Spinger (c). 274-288, 2004
11. Charles Forgy: Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem: Artificial Intelligence, 19, 17-37, 1982
12. Michael K. Smith, Chris Welty, and Deborah L. McGuinness: OWL Web Ontology Language Guide. W3C Recommendation 10 February 2004. (2004)
13. Peter F. Patel-Schneider, Patrick Hayes, and Ian Horrocks: OWL Web Ontology Language Semantics and Abstract Syntax. W3C Recommendation 10 February 2004.
14. Richard Waldinger: Formulation and Validation of the Axiomatic Semantics of OWL. (http://www.ai.sri.com/daml/owl/axiomatic.htm).
15. Raphael Volz, Stefan Decker, Daniel Oberle: Bubo - ImplemenBossamg OWL in rule based systems. (http://www.daml.org/listarchive/joint-committee/att-1254/01-bubo.pdf), 2003