Medusa: 시맨틱 웹 규칙 언어 처리를 위한 확장형 서술 논리 추론기

(Medusa: An Extended DL-Reasoner for SWRL-enabled Ontologies)

김 제 민 † 박 영 택 ††
(Je-Min Kim) (Young-Tack Park)

요 약 현재 온톨로지의 논리적 오류와 개념들 간의 포함 관계를 탐지하는 추론 엔진들이 소개되고 있다. 대부분의 서술 논리 기반 온톨로지 추론 엔진은 태블로 알고리즘을 기반으로 구축되었다. 그러나 태블로 알고리즘 기반의 온톨로지 추론은 인스턴스 추론에 있어서 한계를 보인다. 이에 본 논문에서는 Medusa 시스템을 제안한다. Medusa는 서술 논리로 표현된 온톨로지의 정형화된 의미를 기반으로 시맨틱 웹 규칙 언어(SWRL)를 지원하는 확장된 서술 논리 추론 엔진이다. 대부분의 서술 논리 기반 추론 엔진은 효과적으로 온톨로지 스키마 모델을 추론하지만 인스턴스(Assertional Knowledge) 정보를 추론하기 위한 규칙 기반 추론 기능을 제공하지는 않는다. 이러한 문제를 해결하기 위해서 Medusa는 서술 논리의 추론 방식과 규칙 기반 추론 방식을 동시에 사용한다. 본 논문에서 설명하는 Medusa의 프로토타입은 Protégé API[1]를 사용하여 시맨틱 웹 규칙 언어 추론 엔진과 서술 논리 추론 엔진간의 상호작용을 제어한다.

키워드 : 온톨로지, 태블로 알고리즘, 온톨로지 추론 엔진, 서술 논리, 시맨틱 웹 규칙 언어 추론

Abstract In order to derive hidden information (concept subsumption, concept satisfiability and realization) of OWL ontologies, a number of OWL reasoners have been introduced. Most of the reasoners were implemented to be based on tableau algorithm. However this approach has certain limitation. This paper presents architecture for Medusa. The Medusa is an extended DL-reasoner for SWRL(Semantic Web Rule Language) reasoning under well-founded semantics with ontologies specified in Description Logic. Description logic based ontology reasoners theoretically explore knowledge representation and its reasoning in concept languages. However these logics are not equipped with rule-based reasoning mechanisms for assertional knowledge base; specifically, rule and facts in logic programming, or interaction of rules and facts with terminology. In order to deal with the enriched reasoning, The Medusa provides combining DL-knowledge base and rule based reasoner. The described prototype uses Protégé API[1] for controlling communication with the ontology reasoner.

Key words : Ontology, Tableaux Algorithm, Ontology Reasoner, Description Logic, SWRL Reasoning

· 본 논문은 숭실대학교의 지원을 받았습니다.

† 학생회원 : 숭실대학교 컴퓨터학과
kimjemins@hotmail.com
†† 종신회원 : 숭실대학교 컴퓨터학과 교수
park@ssu.ac.kr

논문접수 : 2008년 10월 29일
심사완료 : 2009년 3월 23일

Copyright©2009 한국정보과학회 : 개인 목적이나 교육 목적인 경우, 이 저작물의 전체 또는 일부에 대한 복사본 혹은 디지털 사본의 제작을 허가합니다. 이 때, 사본은 상업적 수단으로 사용할 수 없으며 첫 페이지에 본 문구와 출처를 반드시 명시해야 합니다. 이 외의 목적으로 복제, 배포, 출판, 전송 등 모든 유형의 사용행위를 하는 경우에 대하여는 사전에 허가를 얻고 비용을 지불해야 합니다.

정보과학회논문지: 소프트웨어 및 응용 제36권 제5호(2009.5)

1. 서 론

차세대 웹 환경인 시맨틱 웹에서 중추적인 역할은 하는 것은 기계가 이해할 수 있도록 개념들을 명확하게 명시하고, 이들 개념들을 공유할 수 있는 형식으로 표현한 온톨로지다. 온톨로지[2]는 표준 언어인 RDF(Resource Description Framework)와 OWL(Web Ontology Language)을 사용하여 특정 도메인 지식에 대한 모델로 구축된다. 결국 시멘틱 웹의 목적은 모델링된 도메인 지식을 바탕으로 이기종 간의 상호연동과 자동화된 작업을 이끌어 내는 것이다.

OWL은 시스템간의 공유가 가능한 용어 정의와 추론을 가능하게 함으로써 온톨로지를 구축하는데 사용되는 가장 일반적인 언어가 되었다. OWL은 서술 논리의 표현 구조를 기반으로(특히 OWL-DL) 만들어졌기 때문에, 서술 논리 기반 추론이 가능하다. 서술 논리는 효과적으로 사람의 지식을 표현하고 추론한다. 서술 논리의 표준 추론 작업은 개념의 논리적 사실 여부를 판단하는 논리적 정당성 검사, 개념간의 계층구조를 유추하는 포함 관계 추론, 개체가 어떤 개념에 속하는지 판단하는 실체화로 구분된다[3].

현재 온톨로지의 논리적 오류와 개념들 간의 포함 관계를 탐지하는 서술 논리 기반 추론 엔진들이 소개되고 있다. 대부분의 서술 논리 기반 온톨로지 추론 엔진은 태블로 알고리즘[4]을 적용하여 온톨로지 스키마에 대해서는 완전한 추론 결과를 보여준다. 그러나 태블로 알고리즘 기반의 온톨로지 추론은 시맨틱 웹 규칙 언어(SWRL)를 통한 온톨로지 인스턴스의 실체화를 판단하는데 있어서는 한계를 갖는다. 이러한 이유 때문에 서술 논리 추론 엔진은 질 높은 추론 결과를 보임에도 불구하고 특정 범위로 한정된 지식의 다양한 추론이 필요한 경우 실용적이지 못하다.

이에 본 논문에서는 Medusa 시스템을 제안한다. Medusa는 서술 논리로 표현된 온톨로지의 정형화된 의미를 기반으로 시맨틱 웹 규칙 언어를 지원하는 확장 서술 논리 추론 엔진이다. 대부분의 서술 논리 기반 추론 엔진은 효과적으로 사람의 지식을 표현하고 온톨로지 스키마 모델을 추론하지만 인스턴스(Assertional Knowledge) 정보를 효과적으로 추론하기 위한 규칙 기반 추론 기능을 제공하지 않는다. 이러한 문제를 해결하기 위해서 Medusa는 서술 논리의 추론 방식과 규칙 기반 추론 방식을 동시에 사용한다. 따라서 서술 논리 형식으로 표현된 지식을 규칙 기반 추론 엔진이 처리할 수 있는 형식으로 변환할 수 있는 모듈이 필요하다. 특히 Medusa는 시맨틱 웹 규칙 언어 추론에 있어서 기존의 확장 서술 논리 추론 엔진과는 달리 전방향 추론과 후방향 추론을 동시에 사용할 수 있는 기능을 제공한다. 따라서 두 추론 방식을 제어하기 위해 시맨틱 웹 규칙 언어 브릿지를 구축하였다.

본 연구를 위해 구현한 시맨틱 웹 규칙 언어 브릿지는 서술 논리 형식으로 정의된 개념, 개체 및 시맨틱 웹 규칙 언어로 작성된 규칙을 JESS가 처리할 수 있는 형식으로 변환하는 동시에 JESS 형식으로 표현된 사실들을 다시 서술 논리 형식으로 변환한다. 시맨틱 웹 규칙 언어 브릿지는 JESS 엔진을 사용하여 시맨틱 웹 규칙 언어로 작성된 규칙을 실행하며 JENA와 Protégé API[1]를 사용하여 구현되었다.

본 논문의 구성은 다음과 같다. 2장에서는 Medusa의 기본 구조를 이해하기 위한 기본 개념을 설명하고, 3장에서는 기존의 확장 서술 논리 추론 엔진과 한계점을 설명하며, 4장에서는 본 연구의 기본 아이디어를 간단한 예를 통해 설명한다. 5장에서는 Medusa의 구조와 사용된 기술에 대해서 설명하고, 6장에서는 기존의 확장 서술 논리 추론 엔진과 Medusa의 추론 성능 비교 실험을 통해 Medusa의 효용성에 대해 평가를 한다.

2. 기본 개념

2.1 OWL과 온톨로지 추론

웹 온톨로지 언어인 OWL[6]은 온톨로지를 구축하는데 사용된다. 온톨로지 모델링 관점에서 볼 때 OWL은 서술 논리가 가지고 있는 많은 논리적 구성과 강하게 일치하고 있다. 서술 논리는 사람이 가지고 있는 지식을 기본적으로 개념(Concept), 관계(Role), 개체(Individual)로 표현하고, 이것을 TBox와 ABox로 나눈다. TBox에는 개념의 계층 관계와 정의(Terminological)부분이 선언되는데, 이것은 온톨로지의 클래스(Class)와 제약(Restriction) 정의 부분이다. ABox에는 개념의 개체(Assertional)부분이 정의되는데, 이것은 온톨로지의 개체(Instance) 정의 부분이다.

OWL은 표현 수준에 따라 OWL-Lite, OWL-DL, OWL-Full 3가지 종류로 나뉜다. 이 중에서 OWL-DL은 SHOIN(D) 수준의 서술 논리와 강하게 일치하고 있다. OWL을 위한 추론 서비스는 일반적으로 서술 논리에 대한 추론 서비스와 같이 입력된 온톨로지가 논리적으로 일관성을 갖는지에 대한 일관성 검사, 주어진 온톨로지의 클래스 C와 D 사이에 의미적 포함 관계(*Class Subsumption*)인 $O \models C \sqsubseteq D$가 존재하는지에 대한 검사, 주어진 온톨로지의 개체 a와 클래스 C사이에 개념적 포함 관계(*Instantiation-a*가 C의 개체)가 존재하는지에 대한 검사로 구성된다.

2.2 SparQL

SparQL 질의 언어는 가장 최근에 추가된 시멘틱 웹의 컴포넌트로 대형 RDF 데이터로부터 유용한 정보를 추출하는 강력한 기능을 제공한다. RDF 그래프는 주어(Subject), 술어(Predicate), 목적어(Object)로 구성된 트리플로 표현되는데, SparQL 질의에 대한 결과 역시 RDF 그래프 형식으로 표현된다.

SparQL 질의어의 구조는 기본 그래프 패턴(Basic Graph Patterns)형식으로서 트리플 집합으로 구성된다. SparQL 질의어를 구성하는 트리플 패턴 중 주어와 목적어는 1개 이상의 변수로 대치되거나 그대로 사용된다. 이때 변수의 사용 조건은 변수로 대체할 정보의 양이 유한해야한다는 것이다. SparQL 질의에 대한 결과는

검색 대상이 되는 RDF로부터 질의문에 존재하는 트리플과 단일화(Unify)를 하여 변수에 변수가 위치하는 트리플 노드를 매핑 함으로써 도출된다.

SparQL 질의문은 SQL처럼 프로젝션 연산(SELECT 명령어), 조인 연산(OPTIONAL 명령어), 유니언 연산(UNION 명령어) 및 제한 연산(FILTER 명령어)과 트리플에 같은 변수를 사용함으로써 보다 복잡하게 구성될 수 있다.

2.3 시맨틱 웹 규칙 언어

시맨틱 웹 규칙 언어[7]는 온톨로지를 구축할 때 규칙을 정의하기 위해 설계된 언어로서 규칙 마크업 언어 중의 하나인 일진/이진 데이터로그 RuleML[8]과 OWL DL 및 OWL Lite를 기반으로 규칙을 정의할 수 있는 용어를 제공한다. 시맨틱 웹 규칙 언어는 온톨로지를 구성하는 개체에 관한 추론(개체간의 관계, 실체화)을 수행하기 위해 온톨로지의 개념과 프로퍼티 및 개체들을 사용하여 혼 절 형식의 규칙을 표현한다.

대부분의 규칙 정의 언어처럼 시맨틱 웹 규칙 언어로 작성된 규칙 역시 전제 조건과 결과 쌍의 형식으로 정의한다. 이를 시맨틱 웹 규칙 언어에서 제공하는 용어로 나타내면 전제 조건은 바디(Body), 결론은 헤드(Head)로 표현된다. 헤드와 바디는 하나 이상의 아톰(Atom)의 논리곱으로 구성된다.

온톨로지를 구성하는 개체들의 시맨틱 웹 규칙 언어 기반 추론은 온톨로지의 개념과 프로퍼티를 중심으로 이행된다. 예를 들어, "사람이 남자와 Sibling 관계를 맺고 있다면 그 사람과 남자는 서로 형제 관계다"라는 규칙이 시맨틱 웹 규칙 언어로 정의됐다고 가정했을 때, 사람, 남자, Sibling 관계, 형제 관계라는 용어는 온톨로지 내의 개념과 프로퍼티에서 참조된다. 즉, 사람과 남자라는 용어는 온톨로지 클래스인 '사람'과 '사람'의 서브 클래스인 '남자'에서 참조되고 Sibling 관계와 형제 관계는 '사람'의 프로퍼티인 'hasSibling'과 'hasBrother'에서 참조된다.

위의 예를 시맨틱 웹 규칙 언어로 표현하면 다음과 같다.

$$\boldsymbol{Person}(?x1) \wedge \boldsymbol{hasSibling}(?x1, ?x2) \wedge$$

$$\boldsymbol{Man}(?x2) \rightarrow \boldsymbol{hasBrother}(?x1, ?x2)$$

3. 관련 연구

서술 논리 추론 엔진은 일관성 검사, 의미적 포함 관계(*Class Subsumption*)에 대한 검사, 개념적 포함 관계 또는 실체화를 수행한다. 그러나 태블로 알고리즘 기반의 서술 논리 추론 엔진은 특정 규칙들이 존재하는 도메인에 대해서 개체간의 관계 추론이나 실체화를 실행하는 것에 제한을 갖는다. 이러한 문제점을 해결하기 위해서 규칙 기반이 가능하도록 서술 논리 추론 엔진을 확장할 필요가 있다. 현재 Pellet과 KAON2의 경우 서술 논리 추론 엔진이 갖는 한계점을 해결하기 위해 시맨틱 웹 규칙 언어와 서술 논리를 동시에 처리할 수 있도록 하고 있다.

Pellet[11]은 DL-safe 규칙을 처리할 수 있도록 기존의 태블로 알고리즘을 확장하였다. 따라서 시맨틱 웹 규칙 언어로 작성된 규칙이 첨부된 온톨로지가 로드될 경우 규칙 부분은 DL-safe 규칙으로 변환되어 추론을 실행한다. 그러나 Pellet은 일부 시맨틱 웹 규칙 언어가 제공하는 용어(익명 클래스, 데이터 타입 프로퍼티, 규칙 정의를 위한 내장 함수)들을 지원하지 못한다. 또한 실험 결과 DL-safe 규칙은 대형보다는 중소형 크기의 온톨로지에 대해 실용적이었다.

KAON2[12]는 일반적인 온톨로지 추론 엔진처럼 태블로 알고리즘을 사용하는 대신 선언 명제 기반 데이터로그 프로그램을 사용하여 지식 베이스를 줄여주는 알고리즘에 의해 구현되었다. KAON2 역시 시맨틱 웹 규칙 언어를 처리 할 수 있도록 DL-safe 규칙을 사용하기 때문에 일부 용어를 지원하지 못한다.

따라서 Pellet과 KAON2는 시맨틱 웹 규칙 언어가 사용하는 익명 클래스, 데이터 타입 프로퍼티, 규칙 정의를 위한 내장 함수등을 지원하지 않기 때문에, 작성된 모든 규칙을 완전히 처리하지 못한다. 이 점은 시맨틱 웹 어플리케이션을 구현하는데 많은 제약을 준다. 때문에 시맨틱 웹 규칙 언어로 작성된 규칙들을 완전히 처리할 수 있는 확장 서술 논리 추론 시스템이 필요하다. 더욱이 Pellet과 KAON2는 규칙 기반 추론에 대해서 전방향 추론만 하고 있다. 이러한 방식은 온톨로지 내에 많은 개체가 존재할 경우 규칙 기반 추론 엔진의 지식 베이스에 공간 복잡성(Spatial Complexity)을 가중 시킬 수 있다. 따라서 시간 복잡성은 높지만 공간 복잡성을 줄일 수 있는 후방향 추론의 적절한 적용이 필요하다.

본 논문에서 제안하는 Medusa는 시맨틱 웹 규칙 언어가 제공하는 OWL 한정 용어와 내장 함수를 전방향 및 후방향 추론이 가능한 JESS 규칙으로 변환함으로써 이 두 추론 방식을 적절하게 제어할 수 있는 기능을 제공하는 시맨틱 웹 규칙 언어 브릿지를 개발 적용하여 Pellet과 KAON2가 갖는 단점을 해결한다. 시맨틱 웹 규칙 언어 브릿지는 5장에서 자세히 설명한다.

4. Medusa 추론 예제

Medusa에 대해 자세한 설명을 하기 전에 본 장에서는 간단한 예제를 통해서 본 논문의 아이디어를 설명한다. 먼저 도메인 온톨로지 내에 다음과 같은 클래스, 클

래스 간의 계층(포함 관계), 인스턴스, 인스턴스 간의 관계 및 시맨틱 웹 규칙 언어로 작성된 규칙이 존재한다고 가정한다.

$Professor \sqsubseteq \tau$, $Student \sqsubseteq \tau$, $Faculty \sqsubseteq \tau$,
$UnderStudent \sqsubseteq Student$, $GradeStudent \sqsubseteq Student$
$\tau \sqsubseteq \exists hasTeaching - .Professor$,
$GradeStudent \sqsubseteq \exists hasTeaching.UnderStudent$
$Park \sqsubseteq Professor$, $Kim \sqsubseteq GradeStudent$,
$Soo \sqsubseteq UnderStudent$,
$Professor(?x) \rightarrow Faculty(?x)$

다음, 도매인 온톨로지에 서술 논리 추론을 실행시키면 아래와 같은 포함 관계 추론 및 실체화 결과가 온톨로지에 추가된다.

$GradeStudent \sqsubseteq Professor$, $Kim \in Professor$

클래스 'GradeStudent'는 서술 논리 추론에 의해 'Professor'의 하위클래스로 분류된다. 따라서 'GradeStudent'의 개체인 'kim'은 'Professor'의 개체로 실체화된다. 다음 온톨로지에 정의된 시맨틱 웹 규칙 언어 형식의 규칙은 규칙 기반 추론 엔진인 JESS에서 처리할 수 있는 규칙의 형태로 변환된다. JESS는 Medusa의 규칙 기반 추론엔진을 구현하기 위해 사용된 자바 기반의 전문가 시스템 쉘이다. 따라서 시맨틱 웹 규칙 언어 형식의 규칙은 본 연구를 위해 구현한 시맨틱 웹 규칙 언어 브릿지를 통해 다음과 같이 JESS 규칙으로 변환된다.

$(defruleRule - 1(Professor(name\ ?x))$
$\Rightarrow$
$(assert(Faculty(name\ ?x)))$
$(assert\ OWLIndividual\ ?x\ Faculty))$

다음 온톨로지에 정의된 모든 클래스와 인스턴스 정보는 JESS가 처리할 수 있는 사실 형태로 변환되어 JESS의 지식 베이스에 저장되어야 한다. 따라서 온톨로지 정보는 시맨틱 웹 규칙 언어 브릿지를 통해 다음과 같은 사실들로 변환된다.

$(owl:Thing(name\ Park))$ $(Professor(name\ Park))$
$(owl:Thing(name\ Kim))$ $(Professor(name\ Kim))$
$(GradeStudent(name\ Kim))$ $(owl:Thing(name\ Soo))$
$(UnderStudent(name\ Soo))$

JESS의 추론엔진은 전방향 추론을 실행하여 새로운 사실을 다음과 같이 추론하여 지식 베이스에 추가한다.

$(Faculty(name\ Park))$ $(Faculty(name\ Kim))$

규칙 기반 추론을 통해 추론된 사실들은 온톨로지 실체화의 결과로써 다시 온톨로지에 반영된다. 따라서 다음과 같은 인스턴스와 인스턴스 개념 정보가 시맨틱 웹 규칙 언어 브릿지를 통해 온톨로지에 추가된다.

$Park \in Faculty$, $Kim \in Faculty$

마지막으로 모든 'Faculty'의 구성원을 검색하기 위해 SparQL 질의문을 입력했을 때,

$SELECT\ ?faculty$
$WHERE\{\langle http://ssu.ac.kr\sharp Faculty\rangle\ \langle rdf::Type\rangle\ ?faculty\}$

다음과 같은 검색 결과를 얻을 수 있다.

$?faculty \Rightarrow Park, Kim$

만약 시맨틱 웹 규칙 언어를 처리하는 규칙 기반 추론을 실행하지 않고, 일반적인 서술 논리 추론만 실행할 경우 위와 같은 검색 결과를 얻을 수 없다. 물론 서술 논리 추론 없이 시맨틱 웹 규칙 언어 추론만 실행하는 경우는 'Faculty'의 구성원을 찾는 질의 결과로서 'park' 하나만 검색 된다. 따라서 특정 도매인 범위 내에서 시맨틱 웹 규칙 언어 추론이 가능한 확장 서술 논리 추론은 시멘틱 검색의 질에 많은 영향을 준다.

다음의 두 번째 예제는 기존의 확장 추론 엔진인 Pellet, KAON2에서는 처리가 불가능하지만 Medusa에서는 처리 가능한 경우를 보인다.

Food $\sqsubseteq \top$, Pizza $\sqsubseteq \top$,
Meat $\sqsubseteq$ Food, Milk $\sqsubseteq$ Food, Vegetable $\sqsubseteq$ Food
Large_Size_Pizza $\sqsubseteq$ Pizza, Regular_Size_Pizza $\sqsubseteq$ Pizza, Meat_Pizza $\sqsubseteq$ Pizza
Meat_Pizza $\equiv \exists$hasTopping.peperoni
Food $\sqsubseteq \exists$hasTopping-.Pizza, $\top \sqsubseteq \exists$same_kind_Food-.Pizza integer $\sqsubseteq$ hasSize-.Pizza
peperoni $\in$ Meat, cheese $\in$ Milk, pimento $\in$ Vegetable Pizza_001 $\in$ Pizza, Pizza_002 $\in$ Pizza
hasSize(Pizza_001, '12'), hasTopping(Pizza_001, peperoni)
hasSize(Pizza_002, '9'), hasTopping(Pizza_002, pimento)
Pizza(?x) $\wedge$ hasSize(?x, ?y) $\wedge$ swrlb:greaterThanOrEqual(?y, 10) $\rightarrow$
Regular_Size_Pizza(?x)
Pizza(?x) $\wedge$ Pizza(?y) $\wedge$ differentFrom(?x, ?y) $\rightarrow$ same_Kind_Food(?x, ?y)

도매인 온톨로지 내에 위와 같은 클래스, 클래스 간의 계층 (포함 관계), 인스턴스, 인스턴스 간의 관계 및 시맨틱 웹 규칙 언어로 작성된 규칙이 존재한다. 주목할 부분은 첫 번째 예제와는 달리 데이터 타입 속성인 "has_size"가 존재하고 "Meat_Pizza"의 "hasTopping" 관계를 제한할 때 익명 클래스의 원소로 정의한 "peperoni"를 사용한 점이다. Medusa는 다음과 같은 추론 결과를 보여준다.

Pizza_001 $\in$ Meat_Pizza, Pizza_001 $\in$ Regular_Size_Pizza
same_kind_Food(Pizza_001, Pizza_002), same_kind_Food(Pizza_002, Pizza_001)

그러나 KAON2는 확장 추론시 익명 클래스, 데이터 타입 속성을 처리하지 못하므로 위의 결과 중 첫 번째

줄에 서술된 두 개의 추론 결과는 유도하지 않는다. 또한 Pellet의 경우 규칙 기반 추론에 있어서 데이터 타입 속성을 처리하는 시맨틱 웹 규칙 언어를 처리하지 못하므로 "Pizza_001 ∈ Regular_Size_Pizza"라는 사실을 유도하지 못한다.

5. 시맨틱 웹 규칙 언어 추론을 위한 확장 서술 논리 추론 엔진

이번 장에서는, Medusa의 구조에 대해 설명한다. Medusa는 시맨틱 웹 규칙 언어 추론을 이행할 수 있는 확장 서술 논리 추론 엔진이다. 따라서 Medusa는 서술 논리의 지식 베이스와 규칙 기반 추론 엔진의 지식 베이스간의 상호 형식 변환을 통해 서술 논리 추론이 갖는 한계를 극복한다.

5.1 시스템 구조

본 논문에서 제안하고 있는 Medusa의 추론 작업 대부분은 서술 논리 추론 엔진이 담당하고, 규칙 기반 추론 엔진은 시맨틱 웹 규칙 언어로 작성된 규칙을 통해 서술 논리 추론만으로 해결되지 않는 온톨로지 실체화만을 담당한다. Medusa의 규칙 기반 추론 엔진은 JESS를 기반으로 구축되었다. 그림 1은 5개의 주요 구성 요소로 이루어진 Medusa의 구조를 보여준다.

1. 서술 논리 추론 엔진: 서술 논리 추론 엔진은 크게 전처리기, TBox 추론 엔진, ABox 추론 엔진으로 구성된다. 온톨로지가 로딩되는 동안 온톨로지를 구성하는 모든 클래스들, 클래스 정의(Restriction)부분과 모든 인스턴스들은 각각 TBox와 ABox에 서술 논리 형식으로 저장된다. 전처리기는 테블로 알고리즘의 효율적인 동작을 위해 지식 베이스에 저장된 온톨로지 내용을 정규화, 흡수화, 내면화 과정을 통해서 최대한 단순화한다. TBox 추론 엔진은 개념(온톨로지 클래스)의 논리적 정당성 검사와 개념 사이에 의미적 포함 관계를 탐지하는 온톨로지 스키마 추론을 실행한다. 논리적 정당성 검사는 개념에 대한 완전 모델을 구축해 나가면서 논리적인 충돌(Clash)이 발생하는지 검사한다. 만약에 모델 구축 과정에서 논리적 충돌을 포함한다면, '비일관적'(*Inconsistent*)이라는 결과를 내고 개념에 대한 모델이 만들어지지 않는다. 반면에 모델 구축 과정에서 어떠한 논리적 충돌도 없다면 개념에 대해 하나의 가능한 모델을 만들고 '일관적'(*Consistent*)이라는 결과를 낸다. ABox 추론 엔진은 온톨로지를 구성하는 개체의 실체화를 실행한다. 실체화는 계층 구조로 표현된 클래스들 중 주어진 인스턴스가 속하는 가장 특정한 클래스를 찾아준다. 따라서 클래스들의 계층 구조를 추론하는 포함 관계 추론을 완전히 종료한 후에 실행한다.

Medusa에서의 서술 논리 추론 엔진의 역할은 클래스 C와 D가 주어지면 이 두 클래스 간의 $O \models C \sqsubseteq D$의 관계가 성립하는지 추론하는 의미적 포함 관계 추론과 클래스 C가 주어졌을 때, C의 인스턴스를 찾는 실체화를 실행하는 것이다.

Medusa의 서술 논리 추론 엔진은 전체 시스템과는 독립적으로 동작하는데, 이는 DIG(DL Implementors Group) 인터페이스[10]를 적용함으로써 가능하다. DIG 인터페이스는 HTTP 기반으로 XML 문법을 사용한 메시지를 통해

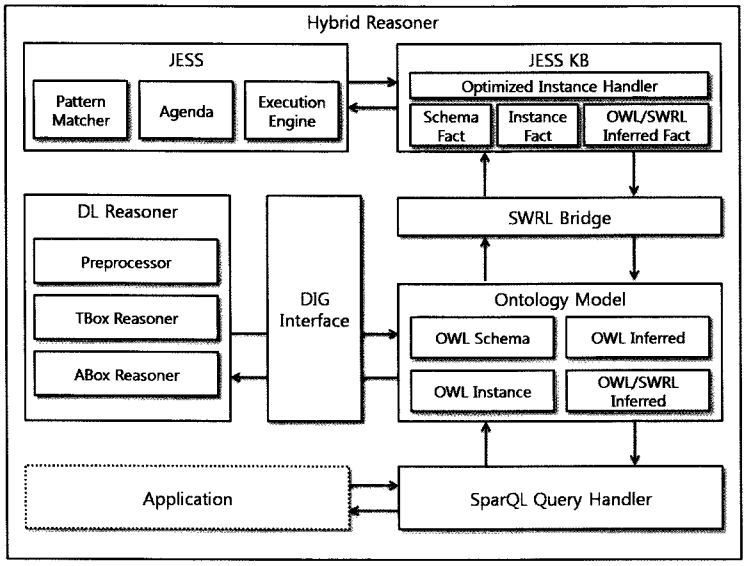

그림 1 Medusa의 구조

어플리케이션과 서술 논리 추론 엔진의 상호작용을 가능하게 해준다. 대부분의 서술 논리 추론 엔진은 DIG 인터페이스를 지원하기 때문에 Medusa에 적용 가능하다.

2. 규칙 기반 추론 엔진: 규칙 기반 추론 엔진은 규칙 베이스, 사실 베이스와 실행 엔진으로 구성된다. 실행 엔진은 사실 베이스 안에 사실들과 규칙의 전제 조건을 구성하는 사실과 매치하여(단일화), 새로운 사실을 추론하고 이를 사실 베이스에 저장한다. Medusa의 규칙 기반 추론 엔진은 JESS 사용하여 구현됐다.

3. 지식 베이스: Medusa의 지식 베이스를 구성하는 사실들은 서술 논리 추론 엔진으로부터 전달되는 온톨로지의 개념(클래스), 프로퍼티, 인스턴스 정보들이 규칙 기반 추론 엔진에서 처리 할 수 있는 형식으로 저장된 것이다. 또한 지식 베이스를 구성하는 규칙들은 시맨틱 웹 규칙 언어로 작성된 규칙이 JESS 규칙 형식으로 변환된 것이다.

4. 온톨로지 모델 핸들러: 온톨로지 모델 핸들러의 기능은 입력된 SparQL 질의에 대해 적절한 응답을 제공하는 것이다. SparQL 질의는 클래스나 인스턴스에 대한 정보를 검색하기 위한 것으로, 보다 정확한 검색 결과를 제공하기 위해서는 서술 논리와 규칙 기반 추론이 같이 실행될 필요가 있다. 예를 들어, 만약 사용자가 특정 클래스의 속하는 모든 인스턴스를 검색한다고 가정했을 때, 온톨로지 모델 핸들러는 먼저 온톨로지에 명시된 인스턴스 정보를 확인하고, 두 번째로 서술 논리 추론 엔진의 실체화 과정을 통해 내포된 인스턴스를 확인하며, 세 번째로 규칙 기반 추론 엔진을 실행하여 추가된 인스턴스 정보를 확인하여 검색 결과로 제공한다. 따라서 더 이상 질의에 해당하는 결과가 지식베이스에 추가되지 않을 때까지 두 번째와 세 번째 단계를 반복한다.

5. 시맨틱 웹 규칙 언어 브릿지: 시맨틱 웹 규칙 언어 브릿지는 JESS를 사용한 규칙 기반 추론 엔진이 시맨틱 웹 규칙 언어로 작성한 규칙을 실행할 수 있도록 지원하는 상호작용 모듈이다. 시맨틱 웹 규칙 언어에 사용되는 내장 함수를 규칙 기반 추론 엔진이 처리할 수 있도록 JESS 규칙으로 변환함으로써 보다 확장된 서술 논리 추론 엔진을 구축하는데 도움을 주는 프로그램으로써 SWRL-JESS tab[5]이 있다. 시맨틱 웹 규칙 언어 브릿지는 SWRL-JESS tab의 기능과 유사하지만, SWRL-JESS tab을 통해 변환된 규칙은 항상 전방향 추론만 실행할 수 있는 형태를 지닌 반면에 시맨틱 웹 규칙 언어 브릿지는 규칙을 후방향 추론이 가능한 형태로 변환하기 위한 특수 프로퍼티 'BK'가 정의되어 있다. 따라서 'BK'가 규칙을 구성하는 전제 조건에 존재하면 시맨틱 웹 규칙 언어 브릿지는 해당 규칙을 후방향 추론 규칙으로 변환한다.

5.2 규칙 기반 추론

일반적으로 규칙 기반 추론 엔진 역시 클래스 간의 포함 관계 추론과 클래스와 인스턴스 간의 실체화 추론이 가능하다. 그러나 규칙 기반의 온톨로지 추론 방식은 정확한 추론 결과만을 제공한다. 즉, 도출될 모든 결과를 추론하지 않으며, 합리적인 시간 안에 추론 결과를 제공하지 못한다. 따라서 효율적으로 포함 관계 및 실체화 결과를 얻기 위해서는 서술 논리 기반의 추론 방식이 필요하다. 대부분의 서술 논리 추론 엔진은 태블로 알고리즘을 사용한다. 이 알고리즘은 합리적인 시간 안에 정확하면서, 도출될 모든 결과를 추론한다. 그러나 서술 논리 추론만으로는 실용적인 온톨로지 추론을 이행하기 어렵다. 예를 들어 온톨로지에 다음과 같은 클래스와 인스턴스가 존재한다면,

$Pet \sqsubseteq \tau$, $Dogs \sqsubseteq Pet$, $Cats \sqsubseteq Pet$, $Person \sqsubseteq \tau$,
$happy \in Dogs$, $kim \in Person$,

$\tau \sqsubseteq \exists hasPet.\tau$, $Person \sqsubseteq \exists owner.Pet$,
$Person \sqsubseteq \exists likePet.Pet$, $likePet \sqsubseteq hasPet$
$likePet(kim, happy)$

일반적으로 태블로 알고리즘 기반 서술 논리 추론 엔진은 아래의 규칙을 처리하지 못한다.

$Person(?x) \wedge hasPet(?x, ?y) \wedge Pet(?y) \rightarrow owner(?x, ?y)$

위의 규칙은 'kim'이 'happy'의 'owner'라는 인스턴스 간의 관계를 보여준다. 따라서 특정 전문 도메인을 표현하는데 있어서, 서술 논리 추론 엔진을 단독으로 사용하는 것은 지식을 바탕으로 실제 컨텍스트를 추론하기에 충분하지가 않다. 이 중에서 대표적으로 중요한 제약 사항은 프로퍼티를 사용하여 다른 클래스의 인스턴스와 관계를 맺는 것이다. 예를 들어 위의 규칙은 'Person'의 인스턴스와 'Pet'의 인스턴스가 'hasPet'관계를 가질 때 'owner'관계를 맺는다는 것을 의미한다. 이러한 정의는 서술 논리 표현만으로는 정의할 수 없다. 따라서 서술 논리 추론 엔진이 갖는 약점을 극복하기 위해서는 규칙 기반 추론 방식을 적용할 필요가 있다.

규칙 기반 추론 엔진은 규칙 베이스, 사실 베이스, 실행 엔진으로 구성된다. 실행 엔진은 사실 베이스 안에 사실들과 규칙 베이스 내에 규칙의 전제 조건을 구성하는 사실과 매치하여(단일화), 새로운 사실을 추론하고 이를 사실 베이스에 저장한다. 시맨틱 웹 규칙 언어로 작성된 규칙의 전제 조건 구문은 클래스와 프로퍼티로 구성되어 인스턴스의 소속이나 관계를 추론할 수 있게 한다. 예를 들어 어떤 규칙의 결과 부분에 인스턴스가 특정 클래스에 소속된다고 정의되어 있으면, 그 규칙의 전제 조건이 만족될 때마다 인스턴스는 해당 클래스의 인스턴스로서 소속되게 된다. 또 다른 예로 특정 프로퍼티로 관계를 맺는 두 인스턴스가 결과 부분에 정의되어

있으면, 전제 조건 부분이 만족될 때 마다 두 인스턴스는 결과에 표현된 프로퍼티 관계를 갖는다.

본 논문에서 제안한 Medusa의 규칙 기반 추론 엔진 부분은 JESS를 사용하여 구축되었다. Pellet과 KAON2는 규칙 기반 추론에 대해서 전방향 추론만 하고 있다. 이러한 방식은 온톨로지 내에 많은 개체가 존재할 경우 규칙 기반 추론 엔진의 지식 베이스에 공간 복잡성(Spatial Complexity)을 가중 시킬 수 있다. 따라서 시간 복잡성은 높지만 공간 복잡성을 줄일 수 있는 후방향 추론의 적절한 적용이 필요하다. JESS는 전방향 및 후방향 추론을 모두 지원한다. 따라서 영역(Domain) 전문가가 시맨틱 웹 규칙을 정의 할 때 특수 프로퍼티 'BK'를 사용하게 되면, 이 규칙은 SWRL 브릿지를 통해서 후방향 추론 규칙으로 변환된다. 예를 들어 다음 규칙은

$$BK \wedge Professor(?x) \rightarrow Faculty(?x)$$

다음과 같이 JESS가 처리 할 수 있는 후방향 추론 규칙으로 변환된다.

$$(do-backward-chaining\ \ Professor)$$
$$(defrule\ \ rule00001$$
$$(need-Professor(name\ ?x))$$
$$\Rightarrow$$
$$(assert(Faculty(name\ ?x)))$$

따라서 규칙 기반 추론이 이행되려면 3 가지의 작업이 진행된다. 먼저, 온톨로지 클래스, 인스턴스, 인스턴스간의 관계를 JESS의 사실 형태로 변환하고, 두 번째로 시맨틱 웹 규칙 언어로 작성된 규칙을 JESS 규칙으로 변환하며, 마지막으로 규칙 기반의 추론을 실행한 후, 추론 결과를 다시 온톨로지에 반영한다.

6. 구현 및 실험

본 장에서는 4장과 5장에서 설명한 Medusa의 성능을 평가한다. 먼저 입력된 SparQL 질의에 대해서 추론을 통한 온톨로지 검색을 실행하는 Medusa 시스템에 대해서 소개하고, Medusa의 정확성과 속도 테스트에 대한 결과를 분석한다. Medusa는 DIG 인터페이스를 사용하기 때문에 다양한 서술 논리 추론 엔진을 그대로 적용할 수 있다.

표 1은 Medusa 성능 실험에 사용된 온톨로지다[13].

표 1 실험에 사용된 OWL 온톨로지

Ontology	Classes	Properties	Individuals	Size
Amino-acid	49	6	120	115K
eukariotic	13	1	11	11K
generations	20	4	7	22K
Ribosome	14	1	10	12k
university	15	5	26	16k
people	62	14	20	48k

이 온톨로지들은 실제 여러 시스템과 어플리케이션에서 사용된 온톨로지다.

그림 2는 입력된 SparQL 질의에 대해서 추론을 통해 검색 결과를 보여주는 Medusa의 인터페이스 화면이다. 이 인터페이스는 실험을 위해 구현된 것이기 때문에 확장 서술 논리 엔진 사이의 상호 작용은 사용자가 제어한다. 사용자는 온톨로지를 추론 엔진이 처리 가능한 형식으로(OWL 모델)로 로드하는 단계, OWL 모델과 시맨틱 웹 규칙 언어로 작성된 규칙을 JESS의 사실과 규칙 형식으로 변환하는 단계, 서술 논리 추론이 실행되는 단계, JESS 규칙 기반 추론이 실행되는 단계, 규칙 기반 추론 결과를 OWL 모델에 반영하는 단계를 직접 제어한다. Medusa 제어는 4개의 버튼으로 이루어진다.

- Load - 지정된 온톨로지를 OWL 모델로 초기화
- Run DL - 서술 논리 추론을 사용하여 실체화 추론을 실행
- Run SWRL - 시맨틱 웹 규칙 언어를 포함한 OWL 모델을 JESS 규칙과 사실로 변환하여, 규칙 기반 추론을 실행한 후, 추론 결과를 OWL 모델에 반영
- Excute - 질의 창에 입력한 SparQL에 대한 검색을 실행

그림 2는 온톨로지가 Medusa에 로드되었을 때 확장 서술 논리 추론에 대한 결과를 보여주고 있다. 인터페이스 중단에 위치한 시각화 창에는 시맨틱 웹 규칙 언어로 작성된 규칙이 적용된 확장 서술 논리 추론을 통해 추론된 개체를 보여준다. 추론이 실행되기 전까지는 '*Boy_5*'는 오직 '*Soccer*'의 개체였다. 그러나 추론 결과로서 '*Boy_5*'는 '*Soccer*'뿐 아니라 '*SoccerPlayer*'의 개체가 된다. 따라서 Medusa의 인터페이스는 '*SoccerPlayer*'를 검색하는 질의가 입력될 때마다 '*Boy_5*'를 검색 결과로 보여준다.

Medusa의 성능 평가를 위해 먼저 각각의 테스트 온톨로지에 대해서 DL-safe 규칙 기반의 확장 서술 논리 추론을 실행하는 Pellet, KAON2와 질의에 대한 응답 속도를 비교 측정하였다. 다음으로 SparQL 질의에 대한 응답 정확도를 검사하였다. Pellet, KAON2와 Medusa 모두 시맨틱 규칙 언어와 SparQL을 지원한다. 본 실험은 펜티엄 CPU(1.73GHz)와 메모리(1.0GB)를 장착한 컴퓨터에서 실행하였으며 Medusa의 서술 논리 추론은 Pellet의 서술 논리 추론 모듈을 사용하였다. Pellet, KAON2와 Medusa 모두 Java로 구현되었으며, Jena의 SparQL 질의 처리 API를 사용한다.

표 2는 각 추론 엔진의 SparQL 질의에 대한 평균 응답 정확도와 평균 응답 시간을 보여준다. 위 실험 결과는 다음과 같이 분석 된다.

1. Pellet은 일부 시맨틱 웹 규칙 언어가 제공하는 용어

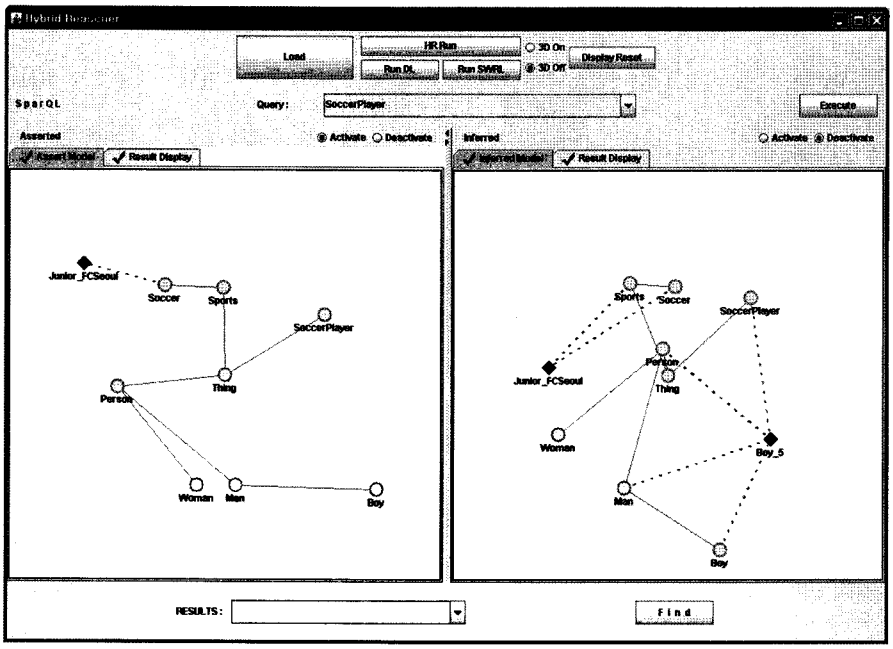

그림 2 Medusa의 인터페이스

표 5 질의에 대한 평균 응답 시간과 평균 정확도

Ontology	Medusa		Pellet		KAON2	
	speed(ms)	precision	speed(ms)	precision	speed(ms)	precision
amino-acid	11350	1.00	83931	0.80	-	0.00
eukariotic	28231	1.00	21068	0.80	-	0.00
generations	-	0.00	-	0.00	-	0.00
ribosome	28565	1.00	-	0.00	850	1.00
university	66375	1.00	92299	1.00	-	0.00
people	20285	1.00	30612	1.00	-	0.00
Average	30961	0.83	56977	0.60	850	0.17

를 처리하지 못함에도 불구하고 SparQL 질의에 대한 응답을 제공하기 위해 서술 논리 추론과 규칙 기반 추론을 동시에 지원한다. 따라서 결과를 산출하기 위해 데이터 타입이나 익명 클래스에 대한 규칙 기반 추론을 요구하는 질의가 입력될 때 마다 Pellet의 검색 결과의 정확도는 Medusa 보다 떨어졌다. 더욱이 Pellet은 처리할 수 없는 질의가 들어올 때 마다 시스템 딜레이 현상 때문에 Medusa에 비해서 검색 속도 역시 떨어졌다. Pellet의 또 다른 취약 부분은 추론을 할 때 힙(Heap) 공간을 너무 많이 사용(*Ribosome Ontology*)한다는 것이다. 또한 현재 Pellet의 서술 논리 추론 엔진은 노미널(Nominal) 개념에 대한 추론이 완벽히 실행하지 못한다. *Generations* 온톨로지의 경우 노미널을 포함하는데, Medusa는 서술 논리 추론 엔진으로 Pellet의 서술 논리 추론 모듈을 사용하므로 이 온톨로지에 대한 추론을 실행하지 못했다.

2. 기본적으로 KAON2 역시 일부 시맨틱 웹 규칙 언어가 제공하는 용어를 처리하지 못할 뿐만 아니라 데이터 타입과 익명 클래스에 대한 서술 논리 추론을 이행하지 못한다. 따라서 익명 클래스와 데이터 타입 속성이 온톨로지에 포함 될 때마다 입력된 질의에 대해 결과를 내지 못한다. 응답 정확도 경우 Medusa는 물론 Pellet에 비해 현저히 낮다. 반면 질의에 대한 응답 속도는 Medusa에 비해 상당히 높았다. 그러나 표 2에서 보듯이 상당수의 온톨로지에 대해 KAON2의 응답 정확도는 0.00이기 때문에 실용성이 떨어진다고 볼 수 있다.

Medusa는 Pellet의 서술 논리 추론 엔진 모듈을 사용하므로 결국 태블로 알고리즘을 기반으로 한다. 이 알고리즘은 논리합 규칙의 분기 선택 문제와 익명 클래스 처리에 있어서 많은 시간을 소모한다. 따라서 Medusa의 응답 시간은 논리합과 익명 클래스로 구성된 클래스의 양에 따라 영향을 많이 받는다. 이 부분에 대해서는 향후 계속 연구가 진행될 것이다.

7. 결 론

서술 논리 기반의 온톨로지 추론은 시맨틱 웹 규칙 언어로 작성된 규칙이 첨부된 온톨로지 인스턴스의 실체화를 판단하는데 있어서는 한계를 갖는다. 이러한 이유 때문에 서술 논리 추론 엔진은 질 높은 추론 결과를 보임에도 불구하고 특정 범위로 한정된 도매인의 상세한 실체화와 개체간 관계 추론이 필요한 경우 실용적이지 못하다. 이에 본 논문에서는 Medusa 시스템을 제안했다. Medusa는 서술 논리로 표현된 온톨로지의 정형화된 의미를 기반으로 시맨틱 웹 규칙 언어 추론을 지원하는 확장 서술 논리 추론 엔진이다. 실용적인 추론을 위해서 Medusa는 서술 논리 추론 방식과 규칙 기반 추론 방식을 동시에 사용한다. 따라서 두 추론 방식을 제어하기 위해 시맨틱 웹 규칙 언어 브릿지를 적용하였다. 특히 Medusa는 시맨틱 웹 규칙 언어 추론에 있어서 기존의 확장 서술 논리 추론 엔진과는 달리 전방향 추론과 후방향 추론 기능을 동시에 제공한다. 시맨틱 웹 규칙 언어 브릿지는 JESS기반의 규칙 기반 추론 엔진이 시맨틱 웹 규칙 언어로 작성된 규칙을 실행할 수 있도록 지원하는 상호작용 모듈이다.

기존의 확장 서술 논리 추론 엔진은 시맨틱 웹 규칙 언어를 정의하는데 사용되는 익명 클래스, 데이터 타입 프로퍼티, 규칙 정의를 위한 내장 함수 등을 지원하지 않기 때문에 시맨틱 웹 규칙 언어로 작성된 규칙을 완전히 처리하지 못한다. 이 점은 시맨틱 웹 어플리케이션을 구현하는데 많은 제약을 준다. 반면 Medusa는 시맨틱 웹 규칙 언어가 갖는 모든 특징을 지원한다.

질의에 대한 평균 응답 정확도와 평균 응답 시간을 분석한 결과 Medusa는 Pellet에 비해서 빠른 응답 속도와 정확한 응답 결과를 보였다. 또한 KAON2에 비해서 속도는 떨어지지만 실용성 높았다.

Medusa는 Pellet의 서술 논리 추론 엔진 모듈을 사용하기 때문에, 확장된 서술 논리 추론의 많은 부분에서 Pellet의 영향을 받는다. 특히 Pellet의 서술 논리 추론 엔진은 노미널(Nominal) 개념에 대한 추론을 완벽히 실행하지 못한다. Medusa 역시 온톨로지에 노미널이 존재할 경우 응답의 정확성이 떨어진다. 따라서 모든 서술 논리 표현에 대한 추론이 가능한 추론 알고리즘과 규칙 기반 추론의 실행 시간을 줄이기 위한 최적화 기법에 대해서 향후 계속 연구가 진행될 것이다.

참 고 문 헌

[1] Holger Knublauch.: Protégé-OWL API Programmer's Guide. http://protege.stanford.edu/plu gins/owl/api/guide.html. September 21, 2006.

[2] T.R. Gruber. A Translation Approach to Portable Ontology Specifications. Knowledge Acqu isition, pp. 199-220, 1993.

[3] R. J. Brachman, R. E. Fikes, and H. J. Levesque. KRYPTON: A functional approach to knowledge representation. *Computer*, pp. 67-73, 1973.

[4] F. Baader and U. Sattler. An overview of tableau algorithms for description logics. *Studia Logica*, pp. 5-40, 2001.

[5] Martin O'Connor, Holger Knublauch, Samson Tu, Mark Musen, Writing Rules for the Semantic Web Using SWRL and Jess, Protégé With Rules workshop collocated with 8th International Protégé Conference, Bethesda, MD, Madrid, 2005.

[6] M. Dean and G. Schreiber. OWL Web Ontology Language Reference W3C Recommendation. http://www.w3.org/tr/owl-ref/. February 2004.

[7] SWRL: http://www.daml.org/rules/proposal/

[8] RuleML: http://www.ruleml.org/

[9] Perez, J., Arenas, M., Gutierrez, C.: The semantics and complexity of SPARQL. In: 5th International Semantic Web Conference (ISWC 2006). (2006).

[10] S. Bechhofer, R. M¨oller, P. Crowther, The DIG description logic interface, in: Proc. Of the Int. Description Logics Workshop (DL 2003), 2003.

[11] Evren Sirin, Bijan Parsia, Bernardo Cuenca Grau, Aditya Kalyanpur and Yarden Katz. Pellet: A practical OWL-DL reasoner, Journal of Web Semantics, 2007.

[12] U. Hustadt, B. Motik, U. Sattler. Reducing SHIQ-Descrption Logic to Disjunctive Datalog Programs. *Proc. of the 9th International Conference on Knowledge Representation and Reasoning (KR2004)*, Whistler, Canada, pp. 152-162, June 2004.

[13] Ontolgy Online, "http://ontologyonline.org/"

김 제 민
정보과학회논문지 : 소프트웨어 및 응용
제 36 권 제 2 호 참조

박 영 택
정보과학회논문지 : 소프트웨어 및 응용
제 36 권 제 2 호 참조