[ISBN 978-89-6211-564-2]

KISTI 지식 총서

추론기술 연구동향과 실용적인 추론

이승우
김 평
정한민
이미경
서동민
성원경

한국과학기술정보연구원

목 차

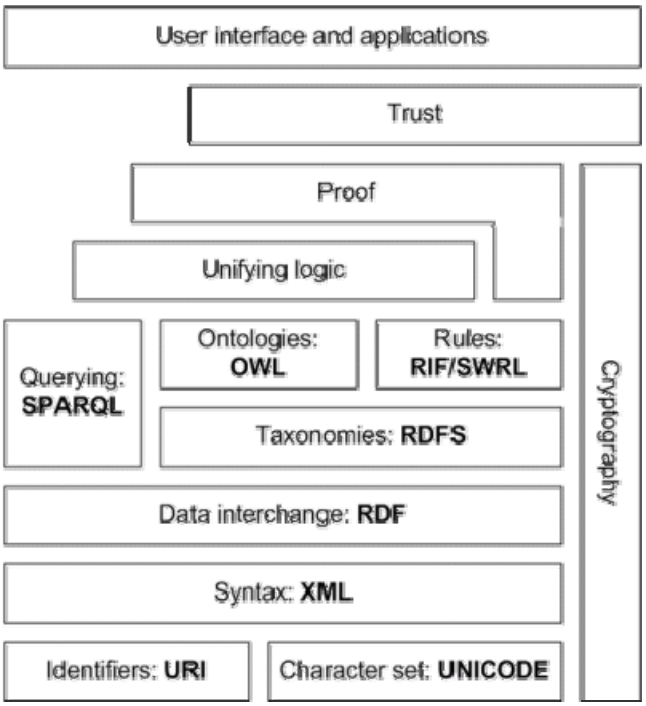
1. 서론

기존 웹은 HTML(Hyper Text Markup Language) 기반으로 문서와 문서를 단순히 연결하는 하이퍼링크로 구성되었다. 이러한 웹을 구성하는 각각의 문서들은 사람이 읽고 이해하기에는 충분하지만 컴퓨터가 스스로 이해하기에는 부적합하다. 또한, 하이퍼링크에 의한 연결이 어떤 의미를 갖는지를 컴퓨터가 이해할 수 있는 수단을 제공하지 못한다. 다시 말해, 웹 상에 상호 연결되어 존재하는 문서들에 대한 자동화된 연계 처리에는 근본적인 한계를 갖는다. 이러한 한계를 극복하기 위해서는 미리 정의된 구조와 의미에 따라 문서의 정보를 기술하고 이를 처리, 가공할 수 있어야 한다. 이러한 목적 하에 출현하게 된 것이 시맨틱 웹[1]이다.

기존의 웹을 '문서의 웹(Web of Documents)'이라 부르는 반면, 시맨틱 웹은 '데이터의 웹(Web of Data)'이라 부른다. 이는 기존의 웹에서는

연결의 대상이 문서인[1] 반면, 시맨틱 웹에서는 연결의 대상이 개별 데이터(자원, 객체)임을 가리킨다. 여기서 개별 데이터는 임의의 것(thing)이 될 수 있다. 사람이 될 수도 있고, 사물이 될 수도 있으며, 특정 사건이 될 수도 있다. 세계 웹 표준화 기구인 W3C 에서는 개별 데이터를 기술하기 위한 표준 언어로서 RDF (Resource Description Framework)를 채택하고 있다. RDF 는 주어와 술어, 목적어의 트리플 (triple)로 구성된 그래프 구조를 갖기 때문에 객체와 객체 간의 관계를 기술하기에 용이하다. 뿐만 아니라, RDF 는 XML 표준과 결합되어 시맨틱 웹 상에서 데이터를 주고 받기 위한 표준으로서 시맨틱 웹의 계층 구조는 [그림 1]의 한 층을 담당한다.

[1] 하나의 문서는 무수히 많은 정보(데이터)를 포함할 수 있다.



[그림 1] 시맨틱 웹 계층 구조[2]

[그림 1]에서 계층의 의미는 상위 계층이 하위 계층을 기반으로 함을 의미한다. 이 계층 구조의 하위 계층들을 보면, 시맨틱 웹이 기존의 웹을 대체하는 것이 아니라, 기존의 웹을 확장하는 것임을 알 수 있다. 즉, 기존의 하이퍼텍스트 웹에서 정의된 URI

[2] 출처: http://obitko.com/tutorials/ontologies-semantic-web/semantic-web-architecture.html

와 UNICODE, XML은 수정 없이 그대로 시맨틱 웹의 기반이 된다. 시맨틱 웹에서는 이 기반 위에 RDF와 RDFS, OWL 등의 언어를 사용하여 온톨로지(ontology)를 표현할 수 있게 한다. RDFS(RDF Schema)는 개념(concept 혹은 class)과 관계(relation 혹은 property, role)를 계층적으로 표현할 수 있는 수단을 제공한다. OWL(Web Ontology Language)은 RDFS의 확장으로, 대응수(cardinality) 제약이나 속성의 값 제약, 이행성(transitivity)과 같은 관계의 특성 제약 등을 사용하여 개념과 관계를 보다 정교하게 기술할 수 있는 수단을 제공한다. 특히, OWL은 기술 논리(Description Logic)[2]를 따르고 있는데, 이런 특성으로 인해 시맨틱 웹의 온톨로지는 추론(reasoning 혹은 inference)을 통해 온톨로지 지식을 확장할 수 있는 논리적 근거를 갖는다. RIF(Rule Interchange Format)와 SWRL(Semantic Web Rule Language)은 기술 논리로 직접 표현될 수 없는 관계를 규칙으로 기술하여 추론을 수행할 수 있도록 하기 위한 확장이다. 이상에서 알 수 있듯이, 시맨틱 웹에서 온톨로

지 기반의 추론은 시맨틱 웹의 중추적인 역할을 담당한다고 말할 수 있다.

본 보고서에서는 우선 시맨틱 웹의 핵심인 온톨로지 기반의 추론이 무엇인지를 전통적인 논리학에서의 추론에 비추어 설명한 후, 최근에 추론 기술이 어떤 방향으로 연구되고 있는지를 최근에 발표된 논문들을 분석하여 살펴보고자 한다. 이와 함께, 시맨틱 웹의 실현을 위해 무엇보다도 절실한 실용적인 추론을 달성하기 위한 하나의 방안으로 Rete 기반 규칙 추론에서 추론 규칙을 동적으로 구체화하는 기법을 소개한다.

2. 온톨로지 기반 추론

시맨틱 웹에서의 온톨로지는, RDF 를 사용해 구조적인 틀을 정의하고, 그 위에 RDFS 와 OWL 을 사용해 의미(semantics)를 정의함으로써 표현된다. 여기서 의미는 개념(concept)과 개념 간의 관계(relation), 개념에 속하는 개체(individual)로 표현된다. 온톨로지 기반의 추론은 온톨로지에 명시적으로 표현된 사실로부터 암묵적으로 내재된 사실을 이끌어 내는 과정이다. 이러한 과정은 내재된 사실의 유형에 따라 다음과 같이 네 가지로 나눠 볼 수 있다.

- 개념 사이의 포함관계(concept subsumption)에 대한 추론
- 관계 사이의 포함관계(relation subsumption)에 대한 추론
- 개체의 개념에 대한 소속(concept membership)에 대한 추론
- 개체와 개체 사이의 관계(relation membership)에 대한 추론

첫 번째와 두 번째는 온톨로지의 구조(schema)에 대한 추론이며,[3] 세 번째와 네 번째는 온톨로지의 개체에 대한 추론이다.[4] 이와 같은 추론을 컴퓨터 상에서 수행하는 방법은 크게 두 가지 유형이 있다. 하나는 기술 논리 기반의 Tableaux 알고리즘을 사용하는 추론 기법[2]이고, 다른 하나는 규칙 기반의 Rete 알고리즘을 사용하는 추론 기법[3],[4]이다. 전자는 온톨로지 기술 언어인 OWL 이 기술 논리를 따르고 있다는 점에서 추론 결과에 대한 정확성(soundness)과 완전성(completeness)을 보장한다. 그러나, 온톨로지 상의 모든 개념과 관계, 개체들 쌍에 대한 충족성 검사(satisfiability check)를 통해서 추론이 이뤄지기 때문에 수 천만 ~ 수십억 RDF 트리플 규모의 대용량 온톨로지에 대한 추론에 있어서는 효율성 측면에서 단점을 갖고 있다. 반면에, 후자는 Rete 알고리즘 같은 효율적인 규칙 적용 기법으로 대용량의 온톨로지에 대한 추론을 처리하는데 있어서 유리하다. 본래의 Rete 알고리즘은 메모리 의존도가 높지만, [4]에

[3] 기술 논리 (Description Logic)에서 사용하는 용어로는 T-Box 추론이라 하며, 여기서 T는 Terminology를 가리킨다.

[4] 기술 논리에서 사용하는 용어로는 A-Box 추론이라 하며, 여기서 A는 Assertion을 가리킨다.

서처럼 하이브리드 Rete 알고리즘을 적용하여 메모리 의존성을 줄일 수 있다. 그러나, 기술 논리와는 표현력의 범위가 다르기 때문에 OWL 로 표현된 온톨로지에 대한 완전한 추론을 지원하지 못하는 단점이 있다. 다시 말해, 규칙 기반 추론 기법은 OWL-DL 수준의 온톨로지의 추론 결과에 대한 정확성은 보장하지만 완전성은 보장하지 못한다.

여기서, 잠시 온톨로지 기반의 추론과 논리학에서 다루는 추론과의 연관성을 살펴 보자. 추론은 논리학에서 연역적 추론(deductive reasoning)과 귀납적 추론(inductive reasoning), 귀추적 추론(abductive reasoning) 등 크게 세 가지로 분류된다.

- P: 모든 사람은 죽는다.

- F: 소크라테스는 사람이다.

- C: 소크라테스는 죽는다.

위와 같은 전제(P: premise)와[5] 사실(F: fact), 결론(C: conclusion)을 예로 설명하자면, 연역적 추론은 전제(P)에 사실(F)을 적용하여 결론(C)을

[5] 전제는 규칙(rule)으로 볼 수 있다. 즉, '모든 사람은 죽는다'는 'X가 사람이면 X는 죽는다'와 같이 규칙으로 표현될 수 있다.

유도(추론)하는 과정을 말한다. 즉, P+F→C 이다. 다음으로, 귀추적 추론은 전제(P)에 결론(C)을 적용하여 사실(F)을 유도(추론)하는 과정이다. 즉, P+C→F 이다. 이 두 가지 추론은 사실(F)과 결론(C)의 위치가 다른 점을 알 수 있다. 즉, 추론의 방향이 서로 반대인 셈이다. 이런 점에서, 온톨로지 기반의 추론 관점에서 볼 때, 연역적 추론은 전방 추론(forward reasoning)에 해당되며, 귀추적 추론은 후방 추론(backward reasoning)에 해당된다고 볼 수 있다. 그러나, 귀추적 추론 자체는 논리적으로 볼 때, 결론(C)를 유도할 수 있는 또 다른 전제와 사실이 가능하기 때문에 논리적 추론 오류를 범할 수 있지만, 후방 추론은 결론(C)의 진위 여부를 판별하기 위해 사실(F)이 참인지를 확인하는 백트래킹(backtracking)을 의미하는 점에서 차이가 있다. 마지막으로, 귀납적 추론은 여러 개의 사실(F)과 결론(C)으로부터 전제(P)를 유도(추론)하는 과정으로, 경험으로부터 얻게 되는 일종의 학습(learning)에 해당한다. 이러한 귀납적 추론은 온톨로지 기반의 추론에서는 군집화(clustering) 기법이나 분류(classification) 기법을 통해 온톨로지 개체의 개념에 대한 소속(concept membership)을

추론하는데 활용된다.

이상에서 설명한 바와 같이, 온톨로지에 대한 전형적인 추론 기술은 기술 논리를 따르는 온톨로지의 표현력과 기술 논리 기반의 효율적인 추론 기법, 규칙 기반의 추론 기법, 전방 추론과 후방 추론 등의 주제를 다룬다. 이러한 주제들의 목적은 최대한 많은 온톨로지 표현력을 지원하면서, 최대한 효율적으로 추론을 수행하고자 하는 것이다.

3. 추론기술의 연구동향

시맨틱 웹의 초기인 2000 년대 초반에서 중반까지 추론 기술 연구는 온톨로지에 대한 전형적인 추론 태스크(standard reasoning task)에 집중되었다. 기술 논리 수준의 온톨로지 표현력을 최대한 지원하면서 대규모의 온톨로지에 대해서도 효율적으로 추론을 수행하는 기법을 개발하기 위해 많은 연구가 있어 왔다. 그러나, 최근 들어서는 시맨틱 웹이 다양한 실제 환경에 적용되기 시작하면서 추론 기술에 대한 연구도 비전형적인 추론 태스크(non-standard reasoning task)로 점차 다변화되는 경향을 보이고 있다.

[표 1] 조사대상 논문의 규모[6]

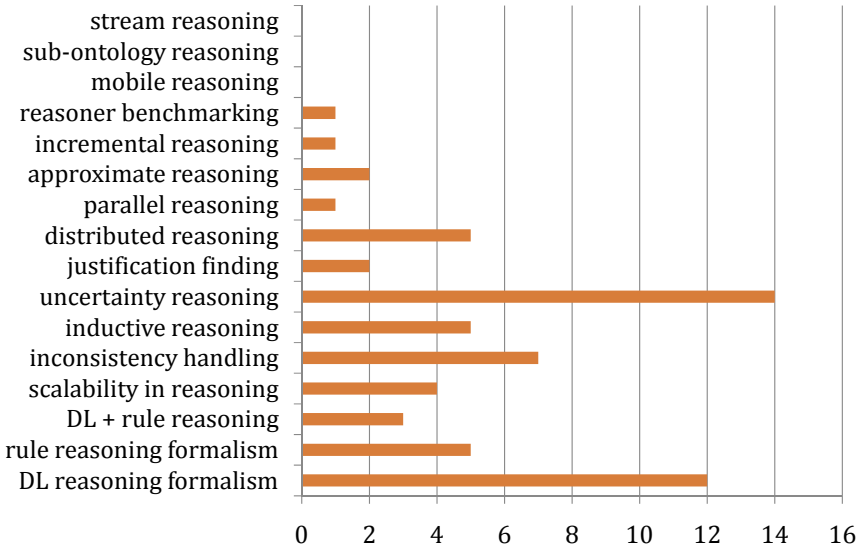
IWSC08	ESWC08	ISWC09	ESWC09	합계
32	13	20	12	77
(9+2+0+1+20)	(6+0+1+0+6)	(8+2+0+0+10)	(5+3+0+0+4)	(28+7+1+1+40)

[6] 괄호 안의 숫자는 순서대로 주요 학술대회 논문, 포스터, 데모, 박사학위 논문, 워크숍 논문들의 건수를 가리킨다.

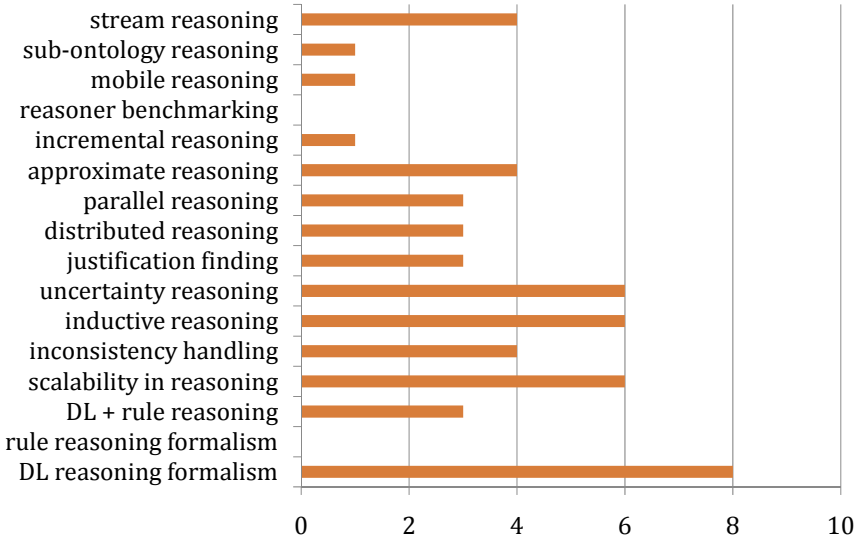
이러한 추론 기술의 최근 연구 동향을 보다 구체적으로 파악하기 위해, 시맨틱 웹을 다루는 대표적인 국제 학술 대회인 ISWC(International Semantic Web Conference)와 ESWC(European Semantic Web Conference)[7]에서 최근 2 년간(2008~2009 년) 발표된 논문을 조사하였다. 조사 대상을 넓히기 위해, 이 학술대회 주요 발표 논문 외에 포스터와 데모, 박사학위 발표 논문들도 포함시켰으며, ISWC 및 ESWC 와 함께 개최된 워크숍 및 다른 학술대회 발표 논문들도 조사 대상에 포함시켰다. 조사 대상 논문들 중에서 추론 관련 기술을 주제로 다루고 있는 논문들을 선정하였으며, 그 규모는 [표 1]과 같다. 이 표에서 보는 바와 같이, 전반적으로 ESWC 보다 ISWC 에서 추론 관련 주제를 다루는 논문이 더 많이 발표된 것으로 나타났다. 이러한 논문들이 구체적으로 어떠한 추론 관련 주제들을 다루고 있는지를 분석하기 위해, 우선 각 논문의 초록과 도입부, 결론부를 읽고 추론 관련 주제를 각 논문 당 5 개 이내로 선정하였다. 다음으로 유사한 주제들을 군집화하고 각 군집에 대표 주제를 할당하였

[7] ESWC는 2010년 학술대회부터는 지역적 제약을 없애기 위해 Extended Semantic Web Conference로 명칭을 변경하였다.

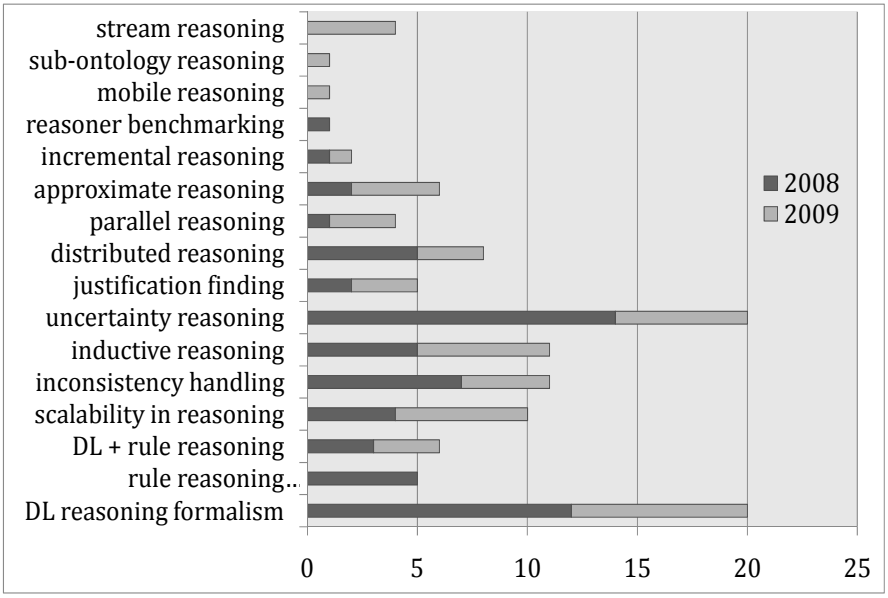
다. 이 과정을 통해 [그림 2]에 보인 바와 같이 14 개의 추론 관련 주제 군집을 얻었고, 각 주제의 연구 활성화 정도를 파악하기 위해, 각 군집별로 대상 논문의 출현 빈도를 조사하였다. [그림 2] ~ [그림 4]는 이렇게 얻은 주제 빈도를 연도별로 구분하여 분석한 결과이다.



[그림 2] 추론 기술 주제 빈도 (2008년)



[그림 3] 추론 기술 주제 빈도 (2009년)



[그림 4] 추론 기술 주제 빈도 (2008년~2009년)

2008 년에는 전형적인 추론 태스크를 다루는 기술 논리(Description Logic: DL) 기반의 추론 방법론(DL reasoning formalism)과 규칙 기반의 추론 방법론(rule reasoning formalism)에 대한 주제가 규모성(scalability)과 함께 여전히 다수로 연구되었다. 한 가지 특이한 점은 추론의 효율성을 위해 기술 논리와 규칙을 결합하는 방법(DL+rule reasoning)에 대한 연구가 출현하였다는 점이다. 또한, 이미 다양한 주제의 비전형적인 추론 태스크를 다루는 연구가 상당히 많이 나타나고 있다. 특히, 실제 응용 환경에서 발생하는 불확실성을 다루는 추론(uncertainty reasoning)과 귀납적 추론(inductive reasoning), 비일관성 처리(inconsistency handling), 다중 온톨로지 환경을 다루는 분산 추론(distributed reasoning) 등의 주제가 두드러지게 출현하고 있음을 알 수 있다. 그 외에도, 온톨로지 디버깅이나 추론 결과에 대한 설명을 다루는 정당성 검증(justification finding)과, 추론의 정확성이나 완전성을 일부 희생하여 추론의 효율성을 높이는 근사 추론(approximate reasoning), 온톨로지 데이터의 지속적인 추가와 삭제를 효율적으로 처리하기 위한 점증적 추론(incremental reasoning), 병렬 처리

기법을 활용하여 추론의 효율성을 높이기 위한 병렬 추론(parallel reasoning) 등을 다루는 연구도 진행되고 있는 것으로 나타났다.

온톨로지 기반 추론 관련 연구 주제가 2009 년에는 좀더 다양해지는 양상을 보였다. 전형적인 추론 태스크 측면에서는 규칙 기반 추론 방법론에 대한 연구는 뜸해진 반면, 기술 논리 기반의 추론 방법론과 기술 논리에 규칙을 결합하는 방법, 대용량 온톨로지를 다루는 추론의 규모성 등에 대한 연구는 여전히 많이 연구되고 있다. 또한, 비전형적인 추론 태스크 측면에서는, 2008 년에 비해, 온톨로지 데이터가 지속적으로 입력되는 상황을 고려한 스트림 추론(stream reasoning)이나 온톨로지 재사용 상황을 고려한 하위 온톨로지 추론(sub-ontology reasoning), 모바일 환경을 고려한 모바일 추론(mobile reasoning) 등 보다 다양한 추론 관련 연구 주제가 출현하고 있음을 확인할 수 있다.

특히, 비전형적인 추론 태스크에 대한 연구 주제의 비율이 2008 년에 비해 상대적으로 늘어난 것이 확인된다. 2008 년에는 전형적인 추론 태스크 주제와 비전형적인 추론 태스크 주제의 출현 비율이 24:38 인 반

면, 2009 년에는 17:36 으로 변화를 보이는데, 이는 시맨틱 웹 기술이 실세계 환경에 적용되는 시도가 점차 증가함에 따라 온톨로지 기반 추론에 관한 연구가 점차 전형적인 추론 태스크에서 비전형적인 추론 태스크로 전이·확산되고 있음을 말해준다.

이제, 최근에 주로 나타나고 있는 기술 논리와 규칙의 결합 및 비전형적인 추론 태스크 관련 연구 주제들이 어떤 내용을 다루는지 대표적 사례와 함께 살펴 보자.

3.1. **기술 논리와 규칙의 결합**

지금까지 기술 논리에 대한 연구와 규칙에 대한 연구는 서로 별개로 진행되어 왔지만, 시맨틱 웹 환경에서는 개별적으로 여러 실질적인 사례들을 다루는데 한계가 있음이 드러났다[6]. OWL 이 기술 논리를 따르고 있기 때문에 기술 논리 기반의 추론 방식은 추론 결과에 대한 정확성과

완전성을 확보할 수 있다. 반면, 규칙 기반의 추론은 완전성을 확보하지는 못하지만, 기술 논리 기반의 추론 방식에 비해 보다 효율적으로 추론을 수행할 수 있다. 그러나, 기술 논리와 규칙의 의미적 기반이 서로 다르기 때문에, 이 둘의 통합 (DL + Rule Reasoning)에는 기술적으로 해결해야 할 부분들이 존재한다. 한 예로, 기술 논리는 열린 세계 가정(open-world assumption)을 따르는 반면, 규칙은 닫힌 세계 가정(closed-world assumption)을 따른다. 이러한 기술적 이슈를 해결하여 기술 논리와 규칙을 결합하려는 연구가 최근의 온톨로지 기반 추론 기술 연구의 한 특징으로 나타나고 있다. 한 예로, [5]에서는 TBox 추론에서의 기술 논리 기반의 추론 알고리즘의 장점과 특정 도메인의 대용량 ABox 추론에서의 규칙 기반의 규모성 측면의 장점을 취하는 방식으로 기술 논리와 규칙을 결합하는 추론 방법을 제시하였다.

3.2. **비일관성 처리**

이론적으로 온톨로지의 지식은 일관성을 가져야 하지만, 시맨틱 웹의 실제 환경에서는 시간에 따른 지식의 변화와 온톨로지의 재사용, 다중 온톨로지 결합 등으로 인해 비일관성 문제가 종종 발생한다. 때로는, 온톨로지의 명시적인 지식에서는 비일관성이 나타나지 않더라도 추론을 통해 얻어지는 묵시적인 지식이 명시화될 때 비일관성이 드러나기도 한다. 최근 들어, 시맨틱 웹이 실제 환경에 적용되는 사례가 점차 늘어남에 따라, 비일관성이 존재하는 온톨로지에 대한 추론 (Inconsistency Handling)을 어떻게 처리할 지에 대한 연구가 중요해졌고 다양한 방법들이 제시되고 있다. 기술 논리 기반의 OWL 은 그 자체로는 비일관성 문제를 처리할 수 없기 때문에, 유사 정통 논리(quasi-classical logic)나 개념 무시 기법(concept forgetting)을 도입하여 OWL 온톨로지에 제한을 가하는 방식으로 비일관성 문제를 다루고 있다[8],[9]. 그 외에, 구문적/의미적 관련성을 계산하거나 우선 순위를 부여하는 방식으로 비일관성 문제를

해결하려는 시도도 있다[7].

3.3. **귀납적 추론**

분산된 소스로부터 데이터가 입력되는 경우처럼, 연역적 추론 방식 (Deductive Reasoning) 보다는 귀납적 방식 (Inductive Reasoning)이 더 적합한 경우도 있다. 입력된 온톨로지 개체에 대한 정보(예를 들면, 어느 클래스에 속하는지)가 부족한 경우, 통계적 방식으로 이를 판단할 수 있는 분류기(classifier) 혹은 군집화기(cluster)를 학습하여, 온톨로지 개체에 대한 부족한 정보를 추론할 수 있다. 이러한 귀납적 추론은 온톨로지 개체를 발굴하거나 어떤 개념에 속하는 개체를 요구하는 질의 응답에 활용될 수 있다[10][11].

3.4. **불확실성 추론**

시맨틱 웹을 실제 환경에 적용하는 데 있어서 발생하는 중요한 문제로, 비일관성 문제 외에 불확실성 문제도 있다. 온톨로지 데이터의 불확실성을 다루기 위한 방법 (Uncertainty Reasoning)으로 주로 제시되는 것이 확률 개념을 온톨로지에 도입하는 것이다[12]-[14]. OWL 의 기반인 전통적 기술 논리에 확률 개념을 추가한 확률적 기술 논리(probabilistic DL 혹은 fuzzy DL)를 정의하고 이를 기반으로 추론을 수행하는 확률적 추론 알고리즘들이 다양하게 고안되어 왔다. 또한 확률적 추론이 대규모 ABox 를 대상으로 실용적으로 적용 가능한지에 대한 연구[15]와 LOD(Linked Open Data)에 확률적 온톨로지 개념을 도입하려는 시도[16], 확률적 개념과 관계를 학습하려는 연구[17]도 나타났다.

3.5. **정당성 검증**

온톨로지에 오류가 있을 때 일반적인 추론을 통해서는 오류가 있는 추론 결과를 집어낼 수는 있지만, 그런 오류를 해결하기 위해서는 잘못된 추론 결과가 왜, 어떤 과정을 통해 발생했는지를 사용자가 이해할 수 있어야 할 텐데, 일반적인 추론 엔진에서는 이에 필요한 정보를 제대로 제공하지 못한다. 이런 이슈를 해결하기 위한 목적으로 추론 결과를 설명하기 위한 여러 가지 정당성 검증 (Finding Justification) 방법들이 제시되었다[18],[19].

3.6. **분산 추론**

시맨틱 웹이 대용량화됨에 따라 추론의 규모성과 성능이 매우 중요한 요소가 되었는데, 분산 추론 (Distributed Reasoning)은 바로 이런 요

구를 충족시키기 위해 도입되었다. 또한, 시맨틱 웹 환경에서는 많은 온톨로지들이 존재하며 이들은 상호 독립적으로 개발되는 경우가 많다. 따라서, 분산 추론은 이런 환경에서 발생할 수 있는 온톨로지 정렬 (ontology alignments) 및 비일관성 문제와 함께 최근에 점차 활발히 연구되는 추세를 보이고 있다[20][21].

3.7. **병렬 추론**

병렬 추론 (Parallel Reasoning)은 시맨틱 웹 데이터의 대용량화에 따른 추론의 규모성 및 성능 문제 해결을 위한 또 하나의 방안으로 제시되고 있다[22][23]. 병렬 추론 방법은 크게 두 가지로 나눠 볼 수 있는데, 하나는 기존의 추론 알고리즘을 그대로 사용하는 대신 온톨로지를 상호 독립적인 모듈로 나누어 각각을 병렬로 추론하는 방법이고, 다른 하나는 추론 알고리즘 자체를 병렬 구조로 고안하는 것이다. 그 외에, 여러 추론

방법의 결과를 앙상블하는 방법도 제시되고 있는데, 이는 속도 향상이 목적이 아니라, 추론 결과가 너무 많을 때 중요한 것을 고르는 것이 목적이다.

3.8. **근사 추론**

근사 추론 (Approximate Reasoning) 또한 시맨틱 웹 데이터의 대용량화에 따른 추론의 효율성을 위해 고안된 추론 방법이다. 다시 말해, 추론의 정확성 혹은 완전성를 일부 희생하여 추론의 속도를 향상시키는 것이 목적이다[24][25]. 따라서, 근사 추론을 적용할 때에는, 추론의 정확성 혹은 완전성의 훼손이 덜 중요한 도메인인지와 사용하는 근사 추론 방법의 질적인 측면을 중요하게 고려해야 한다.

3.9. **스트림 추론**

모바일 환경과 같은 현대의 많은 응용 환경에서 데이터 스트림이 발생한다. 그런데, 기존의 추론 기법들은 대부분 정적인 온톨로지 데이터를 대상으로 하기 때문에, 데이터 스트림처럼 빈번히 바뀌는 동적인 데이터에 대한 추론을 지원하지 못한다. 이에 따라, 데이터 스트림을 시맨틱 웹에 적용하기 위해, 빈번히 바뀌는 동적인 데이터에 대한 추론을 수행하는 방법에 대한 연구가 최근 들어 제시되기 시작하였다[26][27]. 스트림 추론 (Stream Reasoning)은 본질적으로 추론 결과를 점증적으로 유지하는 점증적 추론을 포함하고 있다.

4. 실용적인 추론

추론 기능은 많은 시맨틱 웹 도구의 필수 기능이며, 1979 년 C. Forgy 가 Rete 알고리즘[3][28]을 개발한 이후, 규칙 기반 추론 방식은 가장 많이 쓰이는 추론 기법 중의 하나가 되었다. Rete 알고리즘은 본래 생성 규칙 시스템의 효율성을 위해 고안되었는데, 규칙 기반 온톨로지 추론에도 적합한 알고리즘이다[29]. 여기서, 규칙은 if-then 규칙을 말하며, 이는 왼쪽(LHS)과 오른쪽(RHS)으로 구성된다. LHS 는 다시 하나 이상의 조건으로 구성되는데, 이 조건들이 만족될 때 RHS 가 실행된다. 온톨로지에서 하나의 사실은 주어(subject), 술어(predicate), 목적어(object)로 구성되기 때문에, 각 조건은 일반적으로 트리플 패턴으로 표현된다. 여기서, 트리플 패턴이란 트리플의 주어, 술어, 목적어 중 하나 이상이 변수일 수 있음을 가리킨다. RHS 는 결론으로, 역시 하나 이상의 트리플 패턴으로 구성되는데, RHS 의

모든 변수는 반드시 LHS 에 한번 이상 나타나야 한다. LHS 가 온톨로지의 사실들에 일치하여 그 변수들은 어떤 값으로 할당되면, RHS 는 동일한 변수-값 할당을 적용해 새로운 사실을 생성한다.

이러한 규칙의 예로, W3C 문서에 기술된 함의 규칙들[30][31]을 예로 들 수 있다. 그 문서들은 RDF 및 RDFS, OWL 어휘 의미론에 따른 여러 가지 함의 규칙들을 기술하고 있다. 예를 들어, *rdfs2* 규칙 [*(?x ?a ?y) (?a rdfs:domain ?z)* → *(?x rdf:type ?z)*][8]는 '*rdfs:domain*' 어휘의 의미로부터 도출된 규칙인데, '*rdfs:domain*'은 주어진 속성(property)의 주어(도메인)가 될 수 있는 클래스를 제약한다. 또 다른 예로, *rdfs9* 규칙 [*(?x rdfs:subClassOf ?y) (?z rdf:type ?x)* → *(?z rdf:type ?y)*] 는 두 클래스 사이의 포함관계를 표현하는 '*rdfs:subClassOf* 어휘의 의미를 함의하고 있는 규칙이다.

[8] 화살표(→)왼쪽이 LHS이며 그 오른쪽이 RHS이다. 괄호 안의 세 구성 요소가 트리플 패턴이며, 여기서 물음표(?)로 시작하는 것이 변수이다.

이런 규칙들은 Rete 네트워크에서 있는 그대로 표현될 수 있다. 그러나, 이렇게 하면 Rete의 beta 네트워크에서 비효율성을 야기할 수 있다. 첫 번째 예에서 트리플 패턴 *(?x ?a ?y)* 는 온톨로지 내의 모든 사실(트리플)에 일치되지만, 그 중 극히 일부만이 나머지 트리플 패턴인 *(?a rdfs:domain ?z)* 에 일치된 특정 트리플과 조인될 수 있다. 두 트리플 패턴에 공통인 변수 'a'에 같은 값이 할당되어야 하는 일관성 문제 때문이다. 이러한 일관성을 효율적으로 검사하기 위해 대개는 해쉬(hashing)와 같은 색인 기법이 적용된다. 그러나, 색인 자체도 메모리 공간을 많이 차지한다. 타깃 온톨로지가 점차 커지면, 그러한 색인 크기도 또한 점차 커질 것이며, 결국엔 메모리(RAM)에 유지하기 어려울 수도 있다. 위의 두 번째 예제에서도 유사한 문제가 발생하는데, 이는 모든 클래스 인스턴스 트리플들이 일치될 수 있는 와일드 트리플 패턴 *(?z rdf:type ?x)* 때문이다.

와일드 패턴 규칙은 대규모 온톨로지에 대한 추론의

경우에는 비효율성을 훨씬 더 악화시킬 수도 있다. 여러 대규모 온톨로지 시스템들이 MS SQL 서버나 오라클, MySQL 등의 데이터베이스 엔진을 기반 저장소로 활용하는데[32][33][34], 여기서 트리플은 대개 효율성을 위해 클래스와 속성 별로 나뉘어 저장된다. 추론을 위해 Rete 알고리즘이 사용될 수는 있지만, Rete의 알파 메모리와 베타 메모리를 더 이상 주 메모리에 유지하는 것은 불가능할 것이다. 한 가지 해결책은 알파 메모리와 베타 메모리가 데이터베이스 뷰(view) 기법을 통해 대응하는 테이블들을 참조하도록 하는 것이다[4]. 이 경우에, 트리플 패턴 *(?x ?a ?y)* 의 알파 메모리는 결국 모든 트리플 테이블을 참조해야 하며, 이는 이어지는 조인 노드에서 조인을 수행할 때 불필요한 테이블을 접근하게 하는 비효율성을 야기할 수 있다.

이 장에서는, 온톨로지 스키마를 참조하여 와일드 패턴 규칙을 여러 개의 좀 더 구체화된 비와일드 규칙들로

변환함으로써 Rete에서 조인 연산을 감소시킴으로써 추론의 실용성을 향상시키는 방법을 소개한다. 이 변환은 Rete 프레임워크 내에 포함될 수 있으며, 온톨로지 스키마가 주어질 때 동적으로 처리될 수 있다.

4.1. **Rete 기반 추론과 효율성**

규칙 기반 온톨로지 추론에서 Rete 알고리즘은 패턴 매칭의 효율성 때문에 폭넓게 사용되고 있다. Rete 는 본래 생성 규칙 시스템을 위해 고안되었는데, 그 효율성은 여러 규칙 사이에 공유된 패턴을 반복적으로 일치시키는 것을 피할 수 있음에 기인한다[3][28]. 이 알고리즘은 트리플을 다룰 수 있도록 작업 메모리 요소(WME)를 수정함으로써 규칙 기반의 온톨로지 추론에도 사용될 수 있다.

Rete는 알파와 베타 네트워크의 두 부분으로 구성된다. 알파 네트워크는 일종의 트리 구조인데, 루트를 포함한 중간 노드는 값을 테스트하는 역할을 하고, 알파 노드(메모리)라 불리는 단말 노드는 테스트를 통과한 결과를 저장하는 역할을 한다. 루트에서 알파 노드 사이의 경로는 하나의 트리플 패턴에 대응한다. 알파 노드는 또한 베타 네트워크로의 연결점 역할도 갖는다. 베타 네트워크는 하나 이상의 조인 노드, 베타 노드(메모리), 규칙 노드(생성 노드라고도 불린다)로 구성된다. 조인 노드는 트리플 패턴들 사이의 AND 결합에 대응하며, 변수의 일관성을 유지하며 조인 연산을 처리한다. 조인된 결과는 뒤따르는 베타 노드에 저장된다. 이 베타 노드 아래에는 규칙에서 AND 결합된 트리플 패턴의 수만큼 또 다른 조인 노드와 베타 노드가 뒤따를 수 있다. 결국, 조인 노드 아래에는 추론된 트리플을 생성하는 규칙 노드가 뒤따른다.

Rete에서 가장 시간 소요가 큰 연산은 바로 조인 노드에서

발생하는데, 여기서는 공통된 변수의 일관성 검사를 위해 값을 반복적으로 검색하고 비교해야 하기 때문이다. 이를 효율적으로 수행하기 위해, 대개 해쉬(hashing)나 피라미드 기법(Pyramid technique) 등의 색인 기법이 적용된다[28][35]. 규칙에서의 트리플 패턴의 순서 또한 Rete의 추론 효율성에 영향을 줄 수 있다[35].

Rete는 효율성 측면에서 큰 장점이 있지만, 또한 상당히 많은 메모리 공간을 요구한다는 심각한 단점도 있다. 효율성을 위해 알파 노드와 베타 노드를 주 메모리에 유지하는데, 타깃 온톨로지가 지속적으로 커지면, 이것이 불가능할 수도 있다. 이 문제를 해결하기 위해 하이브리드 Rete 기법이 도입되었다[4].

다음 섹션에서는 Rete 프레임워크 위에서 규칙 기반의 온톨로지 추론의 효율성을 위한 차별적 접근 방법으로 일반적인 규칙을 여러 개의 구체적인 규칙으로 구체화하는 방법을 소개한다. 특히, RDF 의미론[30]과 OWL 의미론[31]에 기술된

와일드 패턴을 갖는 함의 규칙들을 대상으로 하며, 이들 규칙은 온톨로지 스키마를 참조하여 Rete 프레임워크 내에서 여러 개의 특정 규칙들로 변환될 수 있다.

4.2. **동적 구체화 기법**

온톨로지는 주로 RDF 와 RDF/S, OWL 언어로 씌어진다. 이 언어들은 미리 정의된 여러 가지 어휘들로 이루어져 있는데, 이 어휘들의 의미로부터 여러 함의 규칙(entailment rules)들이 유도될 수 있다. 이러한 어휘들은 온톨로지 자체를 기술하기 위해 정의된 것으로, 이들의 함의 규칙들은 포괄적이며 어떤 특정 온톨로지에 대해서도 항상 유효한 것들이라 와일드 패턴을 갖는 경우가 흔하다.

[표 2] RDF 및 OWL 의미론에 기술된 와일드 패턴을 갖는 함의 규칙들

규칙이름	함의 규칙
rdfs2	*(?x ?a ?y) (?a rdfs:domain ?z) → (?x rdf:type ?z)*
rdfs3	*(?x ?a ?y) (?a rdfs:range ?z) → (?y rdf:type ?z)*
rdfs7	*(?x ?a ?y) (?a rdfs:subPropertyOf ?b) → (?x ?b ?y)*
rdfs9	*(?x rdfs:subClassOf ?y) (?z rdf:type ?x) → (?z rdf:type ?y)*
transprop	*(?a rdf:type owl:TransitiveProperty) (?x ?a ?y) (?y ?a ?z) → (?x ?a ?z)*
inverseof	*(?a owl:inverseOf ?b) (?x ?a ?y) → (?y ?b ?x)*

4.2.1. 와일드 패턴과 실마리 패턴

와일드 패턴은 트리플 패턴 중에서 술어부(predicate)가 변수이거나 혹은 술어부가 'rdf:type'이면서 목적어(object)가 변수인 경우를 말한다. 좀더 구체적으로 말하자면, 전자의 경우를 와일드 속성 패턴이라 하는데, 예를 들면, *(?x ?a ?y)* 와 같은

패턴이 해당되며, 후자의 경우를 와일드 클래스 패턴이라 하는데, 예를 들면, *(?z rdf:type ?x)* 와 같은 패턴이 해당된다.

RDF 와 OWL 의미론에 기술된 와일드 패턴을 갖는 함의 규칙의 몇 가지 예를 [표 2]에 보였다. 각 규칙에서 와일드 패턴들은 밑줄로 표시되어 있다. 와일드 패턴을 갖는 규칙 중에는 조인 연산을 필요로 하지 않는 경우도 있지만, [표 2]의 예들은 한번 이상의 조인 연산을 필요로 하는 규칙들이다. 와일드 패턴은 대상 온톨로지의 모든 트리플 혹은 모든 클래스 인스턴스 트리플에 일치될 수 있기 때문에 와일드 패턴을 포함하는 조인 연산은 시간을 매우 많이 소요하게 된다. 따라서, 와일드 패턴은, 온톨로지 스키마를 참조하여 구체화하는 과정을 거침으로써 구체화된 규칙들로 변환될 필요가 있다.

사실, 술어 변수는 온톨로지에 정의된 모든 술어에 일치될 수 있고, 와일드 클래스 패턴의 목적어 변수 (다른 말로, 클래스 변수)는 온톨로지에 정의된 모든 클래스에 일치될 수 있다.

그러한 술어와 클래스는 온톨로지의 스키마로 정의되며, 이 스키마는 추론 규칙이나 온톨로지 인스턴스 (즉, 개별 트리플)에 앞서 입력된다고 가정할 수 있다. 따라서, 술어 변수와 클래스 변수는 온톨로지 스키마를 사용하여, 추론이 수행되기 이전에, 구체화될 수 있는 것이다.

와일드 패턴 규칙, 특히 조인 연산을 필요로 하는 와일드 패턴 규칙들은 와일드 패턴의 술어 변수나 클래스 변수를 제약하는 또 하나의 트리플 패턴을 갖는 경우가 흔히 있다. 이런 트리플 패턴을 실마리 패턴이라 부르는데, rdf2의 경우, *(?a rdfs:domain ?z)* 가 바로 실마리 패턴으로서, 와일드 패턴인 *(?x ?a ?y)* 의 술어 변수 '?a'를 제약한다. 다시 말해, '?a'는 속성(property)이면서 또한 '*rdfs:domain*'의 주어(subject)여야 한다. 다른 예로, rdfs9의 경우, *(?x rdfs:subClassOf ?y)* 가 바로 실마리 패턴에 해당하는데, 와일드 패턴인 *(?z rdf:type ?x)* 의 클래스 변수 '?x'를 제약한다. 다시 말해, '?x'는 클래스이면서,

'*rdfs:subClassOf*의 주어(subject)여야 한다.

4.2.2. 규칙의 구체화 (Materialization of Rules)

이 장에서는 와일드 패턴 규칙이 온톨로지 스키마를 참조하여 구체화되는 과정을 예를 들어서 설명한다. 여기서, 구체화라는 말은 와일드 패턴 규칙을 여러 개의 특정한 (즉, 와일드하지 않은 패턴을 갖는) 규칙들로 변환하는 것을 의미한다. 예를 들어 설명하기 위해, 우선 다음과 같이 정의된 대상 온톨로지 스키마를 가정하자([그림 5]):

<ex:takesCourse, rdfs:domain, ex:Student>
<ex:teacherOf, rdfs:domain, ex:Professor>
<ex:Student, rdfs:subClassOf, ex:Person>
<ex:Professor, rdfs:subClassOf, ex:Person>
<ex:Student, rdf:type, rdfs:Class>
<ex:Professor, rdf:type, rdfs:Class>
<ex:Person, rdf:type, rdfs:Class>
<ex:takesCourse, rdf:type, rdf:Property>
<ex:teacherOf, rdf:type, rdf:Property>
<rdfs:domain, rdf:type, rdf:Property>
<rdf:type, rdf:type, rdf:Property>
<rdfs:subClassOf, rdf:type, rdf:Property>

[그림 5] 온톨로지 스키마의 간단한 예

이와 같은 온톨로지 스키마가 주어질 때, 규칙 rdfs2는 실마리 패턴이 갖는 변수에 대한 제약을 이용하여 다음과 같은 두 개의 특정한 규칙으로 구체화될 수 있다:

rdfs2-1: *(?x ex:takesCourse ?y) (ex:takesCourse rdfs:domain ex:Student)*

→ *(?x rdf:type ex:Student)*

rdfs2-2: *(?x ex:teacherOf ?y) (ex:teacherOf rdfs:domain*

ex:Professor)

→ *(?x rdf:type ex:Professor)*

추가적으로, 이들 구체화된 각각의 규칙들에서 두 번째 트리플 패턴은 더 이상 변수를 갖지 않으며, 주어진 온톨로지에서 항상 만족되는 패턴이다. 따라서, 일반화의 상실 없이, 이들을 구체화된 규칙에서 제거할 수 있으며, 다음과 같은 결과를 얻는다:

rdfs2-1: *(?x ex:takesCourse ?y)* → *(?x rdf:type ex:Student)*

rdfs2-2: *(?x ex:teacherOf ?y)* → *(?x rdf:type ex:Professor)*

이렇게 얻은 두 규칙들은 더 이상 조인 연산을 필요로 하지 않는다. 반면에, 원래의 와일드 패턴 규칙들은 하나의 조인 연산

을 필요로 했었다. 결과적으로, 와일드 패턴의 구체화 과정을 통해 규칙 기반 추론의 효율성이 향상될 수 있게 된 셈이다.

규칙 rdfs9 또한 같은 방식으로 구체화될 수 있으며, 이렇게 얻어진 구체화된 규칙은 다음의 두 규칙이 된다:

rdfs9-1: *(?z rdf:type ex:Student)* → *(?z rdf:type ex:Person)*

rdfs9-2: *(?z rdf:type ex:Professor)* → *(?z rdf:type ex:Person)*

원래의 와일드 패턴 규칙은 조인 연산 하나를 필요로 했지만, 구체화 과정을 거친 이 규칙들은 더 이상 조인 연산을 필요로 하지 않는다.

4.2.3. Rete 프레임워크 내에서의 구체화

와일드 패턴의 구체화 과정은 Rete 프레임워크 내에서 동적으로 구현될 수 있다. Rete 는 규칙 기반의 추론을 위해 널리 사용되는 효율적인 알고리즘이기 때문에 구체화 과정을 Rete 내에서 구현하는 것은 매우 중요하다.

이 과정은 다음과 같이 크게 세 단계로 나눠 볼 수 있다: **(1)Rete 네트워크의 초기 구축 단계**, **(2)온톨로지 스키마를 참조한 동적인 구체화 단계**, **(3)Rete 네트워크 확장 단계**. 이 과정들은, [그림 5]에 있는 간단한 온톨로지 스키마와 규칙 rdfs2 를 사용하여, [그림 6] 및 [그림 7]과 함께 아래에서 차례로 설명한다. 특히 [그림 7]에서는 동적 구체화 기법을 적용한 규칙 기반 추론기의 전체 흐름도를 보여주고 있다.

(1) Rete 네트워크의 초기 구축: 초기에 Rete 네트워크는 주어진 규칙들로부터 생성된다. 모든 비와일드 패턴 규칙들에 대해서는, 규칙 왼쪽(LHS)의 각 트리플 패턴은 보통의

Rete 에서와 같이 루트 노드에서부터 중간의 여러 테스트 노드를 거쳐, 알파 메모리 노드까지의 경로 하나로 변환된다. 반면에, 모든 와일드 패턴 규칙들에 대해서는, 우선 실마리 패턴을 찾는다. 그러나, 간혹 규칙에 실마리 패턴이 없는 경우도 있는데, 이럴 때는 디폴트 실마리 패턴을 사용한다. 좀더 구체적으로 말하자면, 와일드 속성 패턴의 경우에는 *(?a rdf:type rdf:Property)* 를 디폴트 실마리 패턴으로 사용하는데, 여기서 변수 '?a'는 와일드 속성 패턴의 술어 변수에 대응되어야 한다. 마찬가지로, 와일드 클래스 패턴의 경우에는 *(?x rdf:type rdfs:Class)* 를 디폴트 실마리 패턴으로 사용하는데, 여기서 변수 '?x'는 와일드 클래스 패턴의 클래스 변수에 대응되어야 한다.

실마리 패턴을 선정한 후에는, 각 와일드 패턴 규칙은, 규칙의 왼쪽(LHS)에 실마리 패턴만 가지며 오른쪽(RHS)에는 원래의 규칙에서 실마리 패턴을 제거한 규칙을 갖는, 새로운 규칙으로 다시 씌어져야 한다. 예를 들어, [그림 6]에 보인 바와

같이, 규칙 rdfs2 는 실마리 패턴 *(?a rdfs:domain ?z)*을 사용하여 rdfs2′규칙으로 다시 씌어지게 된다.

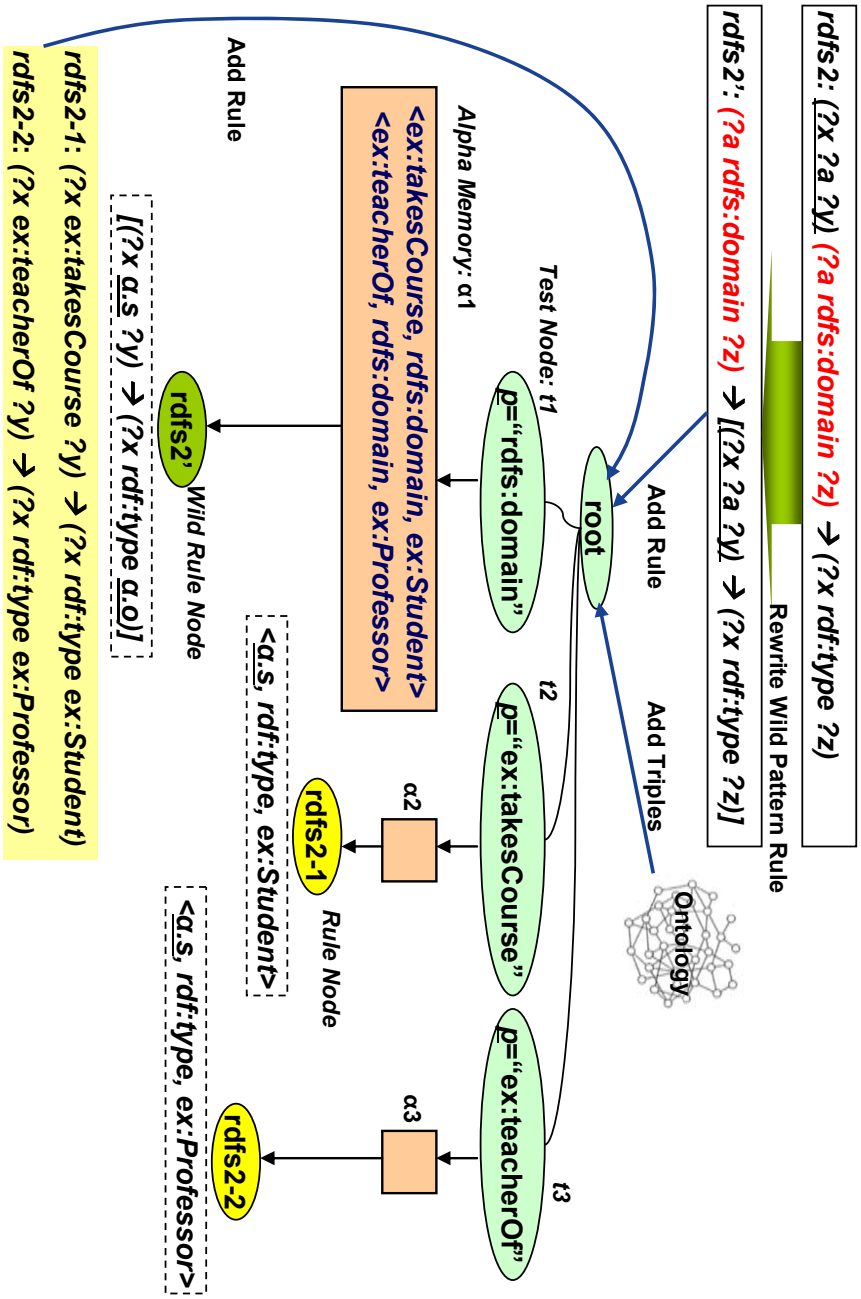
이제, 새로 씌어진 규칙을 Rete 네트워크에 추가하여 일반적인 Rete에서와 같은 방식으로 적절한 테스트 노드와 알파 메모리 노드를 구성하게 되며, 그 아래에 와일드 규칙 노드 (wild rule node)가 뒤따른다. 이 와일드 규칙 노드는 규칙의 오른쪽(RHS)에 대응된다는 점에서 본래 Rete의 규칙 노드와 유사하지만, 규칙 노드는 추론으로 확장된 트리플을 생성하는 반면에 와일드 규칙 노드는 구체화된 규칙을 생성한다는 점에서 분명한 차이점을 갖는다. [그림 6]에 보인 바와 같이, 와일드 규칙 노드 rdfs2′은 구체화를 위한 규칙 템플릿으로 *[(?x α.s y) → (?x rdf:type α.o)]* 를 갖는다. 여기서, *α.s*와 *α.o*는 각각 대응되는 알파 메모리의 주어(subject)와 목적어(object)를 가리킨다.

(2) 온톨로지 스키마를 참조한 동적인 구체화: Rete

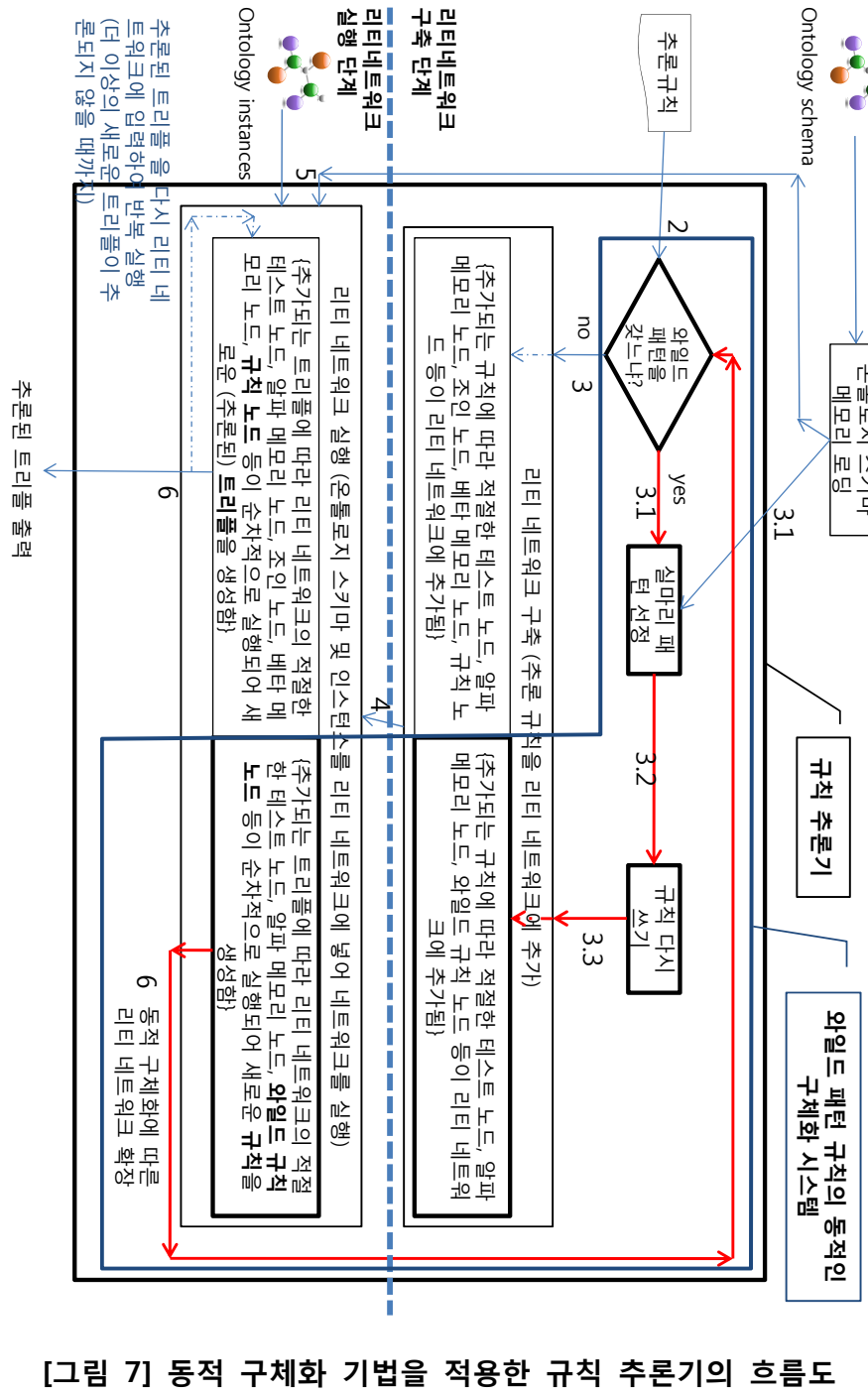
네트워크의 초기 생성 후에는 온톨로지 스키마를 기술하는 트리플들이 Rete 네트워크에 입력된다. 본래의 Rete에서와 마찬가지로, 이 트리플들 중 일부는 Rete의 테스트 노드와 알파 메모리를 차례로 활성화시키고, 결국 와일드 규칙 노드들도 활성화시킬 것이다. 이렇게 와일드 규칙 노드가 활성화될 때, 그 와일드 규칙 노드가 갖고 있는 규칙 템플릿에 의해 특정(즉, 구체화된) 추론 규칙들이 동적으로 생성된다. [그림 6]의 규칙 rdfs2-1과 rdfs2-2가 바로 이런 과정을 통해 구체화된 규칙에 해당된다.

(3) Rete 네트워크 확장: 구체화된 규칙들은 다시 Rete 네트워크에 추가하여 Rete 네트워크를 확장한다. 이 때, 구체화된 규칙이 여전히 와일드 패턴 규칙을 갖고 있다면, 더 이상 와일드 패턴을 갖지 않을 때까지 앞의 두 단계를 반복한다. 더 이상 와일드 패턴을 갖지 않는다면, 구체화된 규칙을 Rete 네트워크에

추가하여 확장한다. [그림 6]에서 보는 바와 같이, t_2와 t_3와 같은 테스트 노드들과, α_2와 α_3과 같은 알파 메모리들, *rdfs2-1*과 *rdfs2-2*와 같은 규칙 노드들 등의 여러 노드들이 구체화의 결과로 Rete 네트워크에 추가된다.



[그림 6] Rete 기반 추론 규칙의 동적인 구체화



[그림 7] 동적 구체화 기법을 적용한 규칙 추론기의 흐름도

4.3. **동적 구체화 기법의 효과**

제안된 구체화 방법의 유효성을 보이기 위해, [표 2]의 함의 규칙들과 LUBM[36]의 온톨로지 스키마를 사용하여 만들어지는 Rete 네트워크에서의 몇 가지 통계 수치를 조사하였다. 우선, 구체화 기법을 적용하지 않은 상태에서 Rete 네트워크를 구성하였을 때의 테스트 노드의 수와 알파 메모리 노드의 수, 조인 노드의 수, 시도된 조인 연산의 수를 세었다. 이 수치들은 [표 3]의 왼쪽 열에 보였다. 또한, Rete 네트워크를 구성할 때 구체화 기법을 적용하였을 때의 동일한 통계 수치도 세었고 이 수치들은 [표 3]의 오른쪽 열에 보였다. 여기서, 계산의 편의를 위해, 추론된 트리플이 Rete 네트워크에 다시 미치는 영향은 배제하였다.

[표 3]을 보면, 구체화 기법에 의해 테스트 노드와 알파

메모리 노드의 수가 꽤 증가하지만, 조인 노드의 수는 효과적으로 줄어듦을 확신할 수 있다. 좀더 자세히 말하자면, 구체화 기법에 의해, 88개의 테스트 노드 (86개의 알파 메모리 노드)가 추가되었지만, 그 중 25는 서로 공유될 수 있는 것이었다. 결과적으로, 71개의 테스트 노드와 69개의 알파 메모리 노드로 늘어난 셈이다. 그러나 조인 연산의 경우에는, 원래 규칙에서는 7개의 조인 노드를 갖지만, 구체화를 적용한 후에는 단지 하나의 조인 노드(transprop 규칙의 경우임)만 남는다.

조인 노드들이 활성화될 때 시도되는 조인 연산의 수를 세기 위해서는, LUBM(1,0)의 온톨로지 인스턴스를 사용하였다. 이는 대략 10만 트리플로 구성되어 있다. 구체화 기법을 적용하지 않았을 때에는, 와일드 속성 패턴이 모든 트리플에 일치될 수 있기 때문에 적어도 100억 번의 조인 연산(transprop 규칙의 경우임)이 시도될 수 있다. 그러나, 구체화 기법을 적용했을 때에는, LUBM(1,0)이 239개의 ub:subOrganizationOf 트리플을

갖기 때문에 단지 57천 번 정도의 조인 연산이 시도될 수 있다. 이 결과로부터, 우리는 와일드 패턴 규칙의 구체화 과정을 통해 규칙 기반의 추론의 성능(속도 측면)이 상당히 개선될 수 있음을 확신한다.

[표 3] 구체화 기법의 효과

	구체화 적용하지 않은 경우	구체화 적용한 경우
# of test nodes	8	71
# of alpha memories	8	69
# of join nodes	7[9]	1
# of join operations tried	10 billions	57121

[9] Rete에서 의미 없이 형식적으로 추가된 더미 조인 노드의 수는 배제하였다.

5. 결론

현재 웹은 사용자 참여를 중시하는 웹 2.0인 소셜 웹에서 의미를 중시하는 시맨틱 웹으로 진화하고 있다. 시맨틱 웹에서 의미(semantics)는 사람과 함께 기계(컴퓨터)가 이해할 수 있음을 가리키는데, 이를 위해 고안된 것이 지식을 표현하는 수단인 온톨로지와 지식을 자동으로 처리하는 수단인 추론 기술이다. RDF와 RDFS, OWL, OWL2 등 온톨로지에 대한 정의가 진화해 가고 그 규모 또한 대용량화되면서 그에 맞게 추론 기술 또한 발전해 가고 있다. 본 고에서는 최근 개최된 시맨틱 웹 관련 국제 학술대회에서 발표된 추론 관련 연구 성과들을 분석하여 최근의 추론 기술이 어떤 방향으로 발전해 가고 있는지를 살펴보았다.

시맨틱 웹의 초기인 2000년대 초반에서 중반까지 추론 기술 연구는 기술 논리 수준의 온톨로지 표현력을 최대한 지원하면서 대규모의

온톨로지에 대해서도 효율적으로 추론을 수행하는 전형적인 추론 태스크(standard reasoning task)에 집중되었다. 그러나, 최근 들어서는 시맨틱 웹이 다양한 실제 환경에 적용되기 시작하면서 추론 기술에 대한 연구도 비전형적인 추론 태스크 (non-standard reasoning task)로 점차 다변화되는 경향을 보이고 있다.

2008년에는 추론의 효율성을 위해 기술논리와 규칙을 결합하는 방법 (DL + Rule Reasoning)에 대한 연구가 출현하였고 다양한 주제의 비전형적인 추론 태스크를 다루는 연구가 상당히 많이 나타났으며, 2009년에는 출현하는 주제가 좀더 다양해지는 양상을 보였다. 특히, 실제 응용 환경에서 발생하는 불확실성을 다루는 추론 (Uncertainty Reasoning)과 귀납적 추론 (Inductive Reasoning), 비일관성 처리 (Inconsistency Handling), 다중 온톨로지 환경을 다루는 분산 추론 (Distributed Reasoning) 등의 주제가 두드러지게 출현하고 있음을 알 수 있었다. 그 외에도, 정당성 검증 (Finding Justification)과 근사 추론 (Approximate

Reasoning), 병렬 추론 (Parallel Reasoning), 스트림 추론 (Stream Reasoning) 등을 다루는 연구도 진행되고 있는 것으로 나타났다. 또한, 2008년에 비해, 2009년에는 비전형적인 추론 태스크에 대한 연구 주제의 비율이 상대적으로 늘어난 것이 확인되었다. 이는 시맨틱 웹 기술이 실세계 환경에 적용되는 시도가 점차 증가함에 따라 온톨로지 기반 추론에 관한 연구가 점차 전형적인 추론 태스크에서 비전형적인 추론 태스크로 전이·확산되고 있음을 말해준다.

온톨로지 추론에 대한 연구가 다양해지고 세분화되고 있음에도 불구하고, 시맨틱 웹의 실현을 앞당기기 위해서는 무엇보다도 중요한 것인 추론 기술의 실용성일 것이다. 현재의 인터넷이 활성화되고 대중의 관심을 갖게 된 것도 바로 실용적인 정보 검색 기술이 함께 제공되었기 때문이다. 이에, 실용적인 온톨로지 추론을 달성하기 위한 하나의 방안으로, 온톨로지

스키마를 참조하여 와일드 패턴 규칙을 동적으로 구체화하는 방법을 제안하였다. RDF 와 OWL 의미론의 많은 함의 규칙들이 조건부에 와일드 패턴을 갖는다. 와일드 패턴은 대상 온톨로지의 모든 트리플에 일치될 수 있기 때문에 규칙 기반의 추론에서 상당한 비효율성을 야기할 수 있다. 더욱이, 그러한 비효율성은 대상 온톨로지의 규모가 커질수록 더욱 심해질 것이다. 그러나, 제안한 구체화 기법을 통해 그러한 비효율성을 피할 수 있으며, 또한 규칙 기반 추론을 위해 널리 사용되는 효율적인 알고리즘인 Rete 프레임워크 내에서 이 구체화 기법을 구현할 수 있다. 구체화 기법의 효과를 측정하기 위해 Rete 네트워크에서 조인 노드의 수가 얼마나 줄어들었는지, 조인 연산의 시도가 얼마나 줄었는지를 세어서, 구체화 기법을 적용하기 전과 비교하였다. 그 결과, 와일드 패턴 규칙의 구체화 과정을 통해 규칙 기반의 추론의 성능(속도 측면)이 상당히 개선될 수 있음을 확신하게 되었다.

[참고 문헌]

[1] T. Berners-Lee, J. Hendler, and O. Lassila, "The Semantic Web", Scientific American Magazine May, 2001.

[2] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, "The Description Logic Handbook: Theory, Implementation and Application (second edition)", Cambridge University Press, 2007.

[3] C. L. Forgy, "Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem", Artificial Intelligence 19(1), 17—37, 1982.

[4] 이승우, 류범종, "메모리 및 DBMS 기반의 하이브리드 Rete 추론", 한국컴퓨터종합학술대회 발표논문집 (KCC2009), 2009.

[5] G. Meditskos and N. Bassiliades, "Combining a DL Reasoner and a Rule Engine for Improving Entailment-Based OWL Reasoning", In Proceedings of ISWC2008, Germany, October 2008.

[6] B. Motik, "Semantics and Reasoning Algorithms for Faithful Integration of Description Logics and Rules", In Proceedings of Web Reasoning and Rule Systems (RR2008), Germany, October 2008.

[7] Z. Huang and F. van Harmelen, "Using Semantic Distances for Reasoning with Inconsistent Ontologies", In Proceedings of ISWC2008, Germany, October 2008.

[8] G. Qi, Y. Wang, P. Haase, and P. Hitzler, "A Forgetting-based Approach for Reasoning with Inconsistent Distributed Ontologies", In Proceedings of the

Workshop on Ontologies: Reasoning and Modularity, Spain, June 2008.

[9] X. Zhang, G. Xiao, and Z. Lin, "A Tableau Algorithm for Handling Inconsistency in OWL", In Proceedings of ESWC2009, Greece, June 2009.

[10] N. Fanizzi, C. d'Amato, and F. Esposito, "Statistical Learning for Inductive Query Answering on OWL Ontologies", In Proceedings of ISWC2008, Germany, October 2008.

[11] G. A. Grimnes, P. Edwards, and Alun Preece, "Instance Based clustering of Semantic Web Resources", In Proceedings of ESWC2008, Spain, June 2008.

[12] F. Bobillo, M. Delgado, and J. Gomez-Romero, "DeLorean: A Reasoner for Fuzzy OWL 1.1", In Proceedings of the Workshop on Uncertainty Reasoning for the Semantic Web, Germany, October 2008.

[13] G. Qi and J. Z. Pan, "Tableau Algorithm for Possibilistic Description Logic ALC", In Proceedings of Web Reasoning and Rule Systems (RR2008), Germany, October 2008.

[14] P. Klinov, "Pronto: a Non-Monotonic Probabilistic Description Logic Reasoner", In Proceedings of ESWC2008, Spain, June 2008.

[15] P. Cimiano, P. Haase, Q. Ji, T. Mailis, G. Stamou, G. Stoilos, T. Tran, and V. Tzouvaras, "Reasoning with Large A-Boxes in Fuzzy Description Logics using DL reasoners: An experimental evaluation", In Proceedings of the Workshop on Advancing Reasoning on the Web: Scalability and Commonsense, Spain, June 2008.

[16] D. Reynolds, "Uncertainty reasoning for linked data", In Proceedings of the Workshop on Uncertainty Reasoning for the Semantic Web, VA, USA,

October 2009.

[17] J. E. O. Luna and F. G. Cozman, "An Algorithm for Learning with Probabilistic Description Logics", In Proceedings of the Workshop on Uncertainty Reasoning for the Semantic Web, VA, USA, October 2009.

[18] M. Horridge, B. Parsia, U. Sattler, and T. Schneider, "Working with Explanations of OWL Entailments", ISWC2009 tutorial, VA, USA, October 2009.

[19] J. Du, G. Qi, and Q. Ji, "Goal-Directed Module Extraction for Explaining OWL DL Entailments", In Proceedings of ISWC2009, VA, USA, October 2009.

[20] A. Schlicht and H. Stuckenschmidt, "Distributed Resolution for ALC - First Results", In Proceedings of the Workshop on Advancing Reasoning on the Web: Scalability and Commonsense, Spain, June 2008.

[21] J. Urbani, S. Kotoulas, E. Oren, and F. van Harmelen, "Scalable Distributed Reasoning Using MapReduce", In Proceedings of ISWC2009, VA, USA, October 2009.

[22] J. Bock, "Parallel Computation Techniques for Ontology Reasoning", In Proceedings of ISWC2008, Germany, October 2008.

[23] J. Weaver and J. A. Hendler, "Parallel Materialization of the Finite RDFS Closure for Hundreds of Millions of Triples", In Proceedings of ISWC2009, VA, USA, October 2009.

[24] S. Rudolph, T. Tserendorj, and P. Hitzler, "What Is Approximate Reasoning?", In Proceedings of Web Reasoning and Rule Systems (RR2008), Germany, October 2008.

[25] T. Tserendorj, S. Rudolph, M. Krötzsch, and P. Hitzler, "Approximate OWL-Reasoning with Screech", In Proceedings of Web Reasoning and Rule Systems (RR2008), Germany, October 2008.

[26] E. D. Valle, S. Ceri, D. Braga, I. Celino, D. Frensel, F. van Harmelen, and G. Unel, "Research Chapters in the area of Stream Reasoning", In Proceedings of the Workshop on Stream Reasoning, Greece, June 2009.

[27] F. Heintz, J. Kvarnström, and P. Doherty, "Stream Reasoning in DyKnow: A Knowledge Processing Middleware System", In Proceedings of the Workshop on Stream Reasoning, Greece, June 2009.

[28] R.B. Doorenbos, "Production Matching for Large Learning Systems", Ph.D. Thesis, Carnegie Mellon University, Pittsburgh, PA (1995)

[29] Y. Katz, K. Clark, and B. Parsia, "Pychinko: a native python rule engine", In: International Python Conference (2005)

[30] RDF Semantics, http://www.w3.org/TR/rdf-mt/

[31] OWL Web Ontology Language Semantics and Abstract Syntax, http://www.w3.org/TR/owl-semantics/

[32] RDFStore, http://rdfstore.sourceforge.net/downloads/

[33] Jena SDB, http://jena.sourceforge.net/SDB/

[34] S. Harris and N. Gibbins, "3store: Efficient Bulk RDF Storage", In Proceedings of the 1st International Workshop on Practical and Scalable Semantic Systems (PSSS'03) (2003)

[35] T. Özacar, Ö. Öztürk, M.O. Ünalir, "Optimizing a Rete-based Inference Engine using a Hybrid Heuristic and Pyramid based Indexes on Ontological Data", Journal of Computers 2(4) (2007)

[36] Y. Guo, Z. Pan, and J. Heflin, "LUBM: A Benchmark for OWL Knowledge Base Systems", Journal of Web Semantics 3(2) (2005)

[ISBN 978-89-6211-564-2]

이승우 • 김 평 • 정한민
이미경 • 서동민 • 성원경

KISTI 지식 총서
추론기술 연구동향과 실용적인 추론

2010년 10월 28일 인쇄
2010년 10월 28일 발행

발 행 처
한국과학기술정보연구원
대전광역시 유성구 어은동 52-11
(우)305-806
전화 : 042-869-1004
등록 : 1991년 2월 12일 제 5-259호

발행인
박 영 서

인쇄처
㈜ 미래미디어