한국지능정보시스템학회논문지 제13권 제3호 2007년 9월(pp.17~29)

차세대 웹 환경에서의 Rete Algorithm을 이용한 정방향 추론엔진 SMART-F 개발*

정균범
연세대학교 정보산업공학과
(leoeternal@yonsei.ac.kr)

송용욱
연세대학교 경영정보학과
(yusong@yonsei.ac.kr)

홍준석
경기대학교 경영정보학과
(junehong@kyonggi.ac.kr)

김우주
연세대학교 정보산업공학과
(wkim@yonsei.ac.kr)

이명진
연세대학교 정보산업공학과
(xml@yonsei.ac.kr)

박지형
연세대학교 정보산업공학과
(zensents@yonsei.ac.kr)

웹 표준 언어인 XML에 기반한 각종 표준들을 바탕으로 소프트웨어 에이전트와의 인터페이스에 초점을 맞추고 있는 차세대 웹에서 소프트웨어 에이전트의 두뇌 역할을 수행하기 위한 추론엔진은 시맨틱 웹(Semantic Web)에서의 규칙 표현을 위한 언어인 SWRL(Semantic Web Rule Language)을 이해할 수 있어야 한다. 본 연구에서는 SWRL을 규칙 표현 방법으로 사용하고, OWL을 사실 표현 방법으로 사용하는 정방향 추론엔진인 SMART-F(SeMantic web Agent Reasoning Tools-Forward chaining inference engine)을 개발하고자 한다.

전통적인 규칙 추론 분야에서는 정방향 추론을 위하여 if-then 형태의 규칙을 네트워크 구조로 변환하여 정방향 규칙 추론의 효율성을 높인 Rete 알고리즘이 많이 사용되고 있다. 이를시맨틱 웹 환경에 적용하기 위하여 SWRL 기반 정방향 추론을 위한 요구 기능을 분석하고, Rete 알고리즘에 도출된 차세대 시맨틱 웹의 요구 기능을 반영한 정방향 추론 알고리즘을 설계하였다. 또한, 유비쿼터스 환경에서의 각종 플랫폼의 독립성과 이식성을 확보하고 기기 간의 성능 차이를 극복할 수 있도록 사실 베이스 및 규칙 베이스의 관리도구와 정방향 추론 엔진 등을Java 컴포넌트로 개발하였으며, 이는 이미 개발된 역방향 추론엔진인 SMART-B와 규칙 베이스 및 사실 베이스를 완벽하게 호환 가능하므로 차세대 웹 환경에서의 지식 활용을 극대화시킬 것이다.

논문접수일 : 2006년 11월 게재확정일 : 2007년 07월 교신저자 : 송용욱

1. 서론

인터넷 상의 웹(Web)이 다양한 분야에서 활용됨에 따라 Blaze Advisor, Exsys, Expertise2go, ILog 등의 추론 엔진들도 웹 기반 시스템으로 상용화되어 고객 지원, 교육, 위험 관리 등의 다양한 분야에서 성공적으로 활용되고 있으나, 이러한 추론 엔진들은 웹을 단순히 사용자 인터페이스 수단으로만 사용하고 있어 웹을 통한 인간 사용자에 대한 서비스만을 목표로 하고 있다. 차세대 웹은 XML과 XML 기반 각종 표준들을 바탕으로 소프트웨어 에이전트와의 인터페이스에 초점을 맞추

* 본 연구는 2006년도 정보통신부 선도기술개발사업 "차세대 웹을 위한 시맨틱 서비스 에이전트 기술 개발"의 일환으로 KT의 지원을 받아 수행되었음.

어 나가고 있으며, 새로운 웹 표준을 활용하여 사용자의 편의성을 증대시킨 비즈니스 모델을 실현해나가기 위하여검색 및 협상 등의 소프트웨어 에이전트에 대한 고차원의 요구들이 증가하고 있다. 이러한 소프트웨어 에이전트들이 인간과의 원활한 정보 교환 뿐만 아니라 스스로 정보를 이해하고 활용하기 위한 수단으로 RDF(Resource Description Framework), RDF-Schema, OWL(Web Ontology Language) 등의 표준을 시맨틱 웹이 등장하였다. 또한 데이터의 자동적인 교환과 재사용을 위한 의미적 처리를 가능하게 하는 언어인 RDF, RDF-Schema, OWL 이외에 if-then 규칙 형태의 지식을 표현하고 활용하기 위한 언어로 OWL과 RuleML (Rule Markup Language)을 조합한 SWRL (Semantic Web Rule Language)과 같은 언어도 개발되어지고 있다.

본 연구에서는 이상과 같은 웹 환경하에서 소프트웨어 에이전트의 핵심적인 기능의 하나인 추론 기능을 SWRL을 규칙 표현 방법으로 사용하고 RDF를 사실 표현 방법으로 사용하는 정방향 추론 엔진인 SMART-F(SeMantic web Agent Reasoning Tools-Forward chaining inference engine)로 개발하고자 한다. 정방향 추론은 전통적으로 if-then 형태의 규칙을 트리구조로 변환하여 규칙의 내용뿐만 아니라 추론의 진행 상태까지 저장하여 추론을 관리할 수 있게 해주는 Rete 알고리즘이 주로 사용되어 왔다[9, 12]. 이러한 방법론을 시맨틱 웹 환경에 적용하기 위하여 SWRL 기반 정방향 추론을 위해 분석된 차세대 시맨틱 웹의 요구 기능을 반영한 정방향 추론 알고리즘을 설계하였다. 또한, 유비쿼터스 환경에서의 각종 플랫폼의 독립성과 이식성을 확보하고 기기 간의 성능 차이를 극복할 수 있도록 사실 베이스 및 규칙 베이스의 관리도구와 정방향 추론 엔진 등을 Java 프로그래밍 언어를 이용하여 단위 컴포넌트의 형태로 개발한다.

본 논문은 다음과 같이 구성되어 있다. 제 2장에서는 시맨틱 웹의 개념과 추론을 위한 사실베이스 및 규칙 베이스 관점에서 RDF, SWRL 등의 의미를 설명한다. 제 3장에서는 정방향 추론과 Rete 알고리즘의 기초 개념과 함께 새롭게 설계된 시맨틱 웹 기반 정방향 추론 알고리즘을 설명하며, 제 4장에서는 시맨틱 웹 기반 정방향 추론 엔진인 SMART-F의 구조와 개발에 따른 핵심 이슈들을 설명한다. 그리고 제 5장에서는 개발한 SMART-F와 다른 추론 엔진들간의 성능 비교 테스트 과정과 결과를 비교하였다.

2. 시맨틱 웹에서의 데이터와 지식의 표현

시맨틱 웹은 "컴퓨터가 정보의 의미를 이해하고 의미를 조작할 수 있는 웹"으로 정의된다[3]. 기존의 웹 언어인 HTML이 웹 상에서 문서의 표현은 지원하지만 문서간의 관계나 의미 등을 표현할 수가 없다는 단점을 극복하기 위하여 메타데이터의 개념을 통해 웹 문서에 시맨틱 정보를 추가하고, 이를 이용하여 소프트웨어 에이전트나 프로그램 등이 문서의 의미를 자동으로 추출할 수 있는 웹 환경으로 발전하게 된 것이 바로 시맨틱 웹이다.

시맨틱 웹의 기본적인 구성요소로는 메타데이터의 문법적규약을 기술하고 있는 RDF와 의미적 규약을 기술하고 있는 OWL이 있다. RDF의 기본적 패턴은 모든 데이터를 표현하고자 하는 해당 데이터의 객체인 Resource, 객체의 Property 및 그 Property가 가지는 Value 값의 트리플(Triples) 형태로 구조화이다(RDF Primer). 이러한 구조 위

에 해당 데이터의 의미를 표현하기 위한 다양한 표현 방식을 지원하는 언어로 OWL을 사용한다. 이에 추가적으로 if-then 형태의 규칙이 갖는 의미를 표현하기 위한 언어로 시맨틱 웹 규칙 언어인 SWRL이 있다. SWRL은 단위 트리플인 Atom으로 이루어진 조건부 body 부분과 결론부 head 부분으로 구성되며, 이러한 규칙을 이용한 추론은 사실베이스와 규칙 베이스를 기반으로 일치하는 트리플 패턴을 찾아내는 방식으로 수행된다.

2.1 데이터 언어 RDF와 OWL

시맨틱 웹에서 메타데이터 표현, 저장 및 관리를 위해 사용하는 가장 기본적인 문서 표현 언어로는 리소스 기술 언어인 RDF와 온톨로지 기술 언어인 OWL이 있다. RDF의 목표는 표준화되고 상호운용성 있는 메타데이터의 처리이며, 이를 위하여구조화된 XML 언어를 이용한다. 또한 특정 도메인을 위한 어플리케이션에서 사용되는 특정한 어휘를 정의하거나 어휘간의 의미적인 관계를 정의하기 위하여 RDF 스키마를 사용한다[1]. RDF 스키마는 XML 스키마처럼 XML 문서의 문법적 유효성을 검증하는 것이 아니라, RDF 문서에 부가적인 정보를 제공하는 목적으로 사용되는 것이다.

RDF의 문법은 기본적으로 XML 문법을 따르고 있으며, 객체지향적 접근 방식의 개념을 따라 RDF 데이터 모델에서 기술되는 URI를 갖는 모든 객체를 가리키는 Resource를 주어(subject)로, Resource의 속성명을 의미하는 Property를 술어(predicate)으로, 속성값에 해당하는 Value를 목적어(object)로 갖는 트리플 구조이다. RDF 데이터 모델은 RDF 모델과 구문에 대한 네임스페이스를 정의하고, RDF 스키마의 정보를 갖는 네임스페이스를 정의하는 것으로 시작되며, 본문에 URI를 갖는 자원과 속성, 속성값등을 기술한다. RDF에서는 독자성을 갖고 있는 모든 개체는 리소스로 사용되며, URI는 이것을 설명하기 위한 것이 아니라 단순히 고유성을 부여하기 위한 인식자로 사용된다.

RDF와는 달리 웹 온톨로지를 기술하기 위한 언어인 OWL은 RDF와 RDF 스키마보다 풍부한 어휘와 형식적 의미론을 포함하고 있다. 즉, RDF 스키마에서는 표현할 수 없었던 클래스간의 관계(교집합, 합집합) 뿐만 아니라 동치성(Equality)과 비동치성, 차수(Cardinality), 속성간의 역관계 등의 세부적인 어휘 관계를 기술할 수 있다. OWL이 시맨틱 웹을 구성하는데 중요기술로 인식되는 이유는 분산된 자원들의 온톨로지를 쉽게 병합(Merge)하거나 다시 분리할 수 있어서 개방적 환경에 매우 적합하다는 것이다. 이러한 OWL은 표현력이 서로 다른 세 개의 하위 언어로 구성된다.

첫 번째, OWL Lite는 시소러스 개념에 접근이 용이하고 단순성을 보강하여 웹 응용에 용이한 온톨로지 언어로 클래스 분류 계층과 간단한 제약사항 표현을 필요로 하는 사용자들을 위한 언어이다. 두번째로 OWL DL은 OWL Lite보다 좀 더 논리적인 표현을 위한 온톨로지 언어로 계산학적 완전성(Computational Completeness)과 결정가능성(Decidability)을 유지하면서 최대의 표현력을 활용하고자 하는 사용자에게 적합하다. 마지막, OWL Full은 OWL Lite와 OWL DL의 모든 기능을 포함하는 관계이며 유효성 및 호환성에 있어서 가장 완벽하여 표현력이 가장 풍부한 언어로 계산학적인 어떤 보장 없이 최대의 표현력과 RDF의 유연한 문법을 모두 활용하고자 하는 사용자에게 적합하다. OWL Full은 RDF의 자유로운 구문을 모두 허용한다.

2.2 규칙 언어 SWRL

SWRL은 OWL의 일종인 OWL DL 및 OWL Lite와 RuleML의 하부 언어인 Unary/Binary Datalog RuleML을 통합하여 규칙 표현 및 저장관리를 위한 언어이다[13]. 규칙은 기본적으로 조건(if)과 결론(then) 간의 관계를 표현하는 것으로, SWRL을 통하여 OWL Axiom을 확장한 유사 Horn 규칙을 OWL 기반의 지식베이스와 통합하여 추론에 활용하게 된다.

규칙 표현에서 조건에 해당하는 부분은 body로, 결론에 해당하는 부분은 head로 표현하며, Horn Logic의 구조에 따라 body는 여러 개가 가능하나 head는 오직 한 번만 가능하다. SWRL의 기본적인 구조는 변수 선언, head(규칙에서의 then 부분), body(규칙에서의 if part)의 세 부분으로 나뉠 수 있다. 그리고 head와 body는 Atom들의 RDF List 구조로, 사용되는 Atom은 크게 아래의 7가지로 분류될 수 있다.

- swrl : ClassAtom
- swrl : DataRangeAtom
- swrl : DatavaluedPropertyAtom
- swrl : DifferentIndividualsAtom
- swrl : IndividualPropertyAtom
- swrl : SameIndividualAtom
- swrl : BuiltInAtom

이 중에서 BuiltIn을 제외한 6가지는 모두 RDF의 트리플 데이터를 표현하기 위한 수단이며, Built-In은 e데이터 계산을 위한 약 70여 가지의 내장 함수를 제공한다[13]. 이러한 내장 함수는 크게 comparisons, math, Boolean values, strings, date/time/duration, URIs, Lists 등의 여섯 가지로 구분되며, 각각은 세부적인 기능별 내장 함수를 포함하고 있다. 이러한 내장함수는 규칙 표현에서 계산식을 이용한 조건 표현의 기반 구조를 제공함으로써 SWRL의 표현력을 증가시킨다.

3. 시맨틱 웹 기반 정방향 추론

시맨틱 웹 기반 추론에는 정방향 추론과 역방향 추론이 있다. 이 중 시맨틱 웹 기반 정방향 추론은 트리플로 표현된 사실(Fact)들과 트리플을 조건절 또는 결론절로 갖는 SWRL 기반 규칙들을 조합하여 새로운 사실 트리플을 도출해내는 것이다. 이때, 규칙의 조건절 또는 결론절은 변수를 가질 수 있으며 이를 감안한 Matching 및 Variable binding이 이루어져야 한다. 또한, SWRL은 내장함수를 표현하며, 내장 함수의 인수(Argument)에도 변수가 있을 수 있으므로 이를 감안하여야 한다.

기존에 나와 있는 시맨틱 웹 기반 추론 엔진으로는 Jena, Jess, Bossam등이 있다. Jena의 reasoner는 RDF 그래프의 규칙 기반 추론과 정방향, 역방향 추론을 제공하고 있다. Jena의 정방향 추론 엔진에서도 Rete 알고리즘을 사용하고 있다. Jess는 전부 Java언어를 통해 구현된 Rule 엔진이다. Jess에서도 정방향과 역방향 추론을 모두 지원하며, 정방향 추론의 알고리즘은 Rete 알고리즘을 사용한다. Bossam은 정방향 추론 엔진으로써, 역시 Rete 알고리즘을 사용하며, OWL 추론, URI referencing, 쿼리 프로세싱, SWRL reasoning 등을 지원한다.

본 절에서는 정방향 추론을 위한 기존의 Rete 알고리즘을 트리플 및 내장함수를 고려하여 변형한 수정 Rete 알고리즘을 소개하도록 한다. 수정 Rete 알고리즘은 다음과 같이 크게 4단계로 이루어져 있다.

Step 1 : 규칙 베이스를 컴파일하여 네트워크 형태로 표현한다.

Step 2 : 컴파일된 규칙 베이스와 사실베이스를 조합하여 초기 상충집합(Conflict set)을 만든다.

Step 3 : 상충집합이 공집합이면 Stop. 아니면 Step 4)로 간다.

Step 4 : 상충집합의 한 요소를 꺼내어 컴파일된 규칙 베이스에 적용한다. 적용 결과 새로운 상충집합 요소(들)이 만들어져 상충집합과 사실베이스에 추가된다. Step 3)으로 간다.

3.1 규칙 베이스 컴파일

규칙베이스의 컴파일은 Rete 알고리즘의 핵심 부분으로 규칙의 조건절과 사실 간의 Match 반복 횟수를 최소화함으로써 추론속도를 극대화하기 위한 것이다. 컴파일된 규칙들은 네트워크 형태로 표현되며, 네트워크의 노드들은다음과 같은 4가지 종류의 노드들로 구분된다.

3.1.1 1-입력 노드(One-input node)

1-입력 노드는조건절 트리플의 Subject, Predicate, Object 중 하나의 값을 특정 상수값 또는 같은 트리플 내의 동일한 이름의 변수값과 비교(Match)하기 위한 조건을 나타내는 노드이다. 따라서, 1-입력 노드는 "상수형 1-입력 노드"와 "변수형 1-입력 노드"로 다시 구분된다. 예를 들어, (?x P ?x)라는 트리플에서는 상수 P에 대한 상수형 1-입력 노드 1개와 변수 ?x에 대한 변수형 1-입력노드 1개 등 총 2개의 1-입력노드가 만들어진다.

3.1.2 2-입력 노드(Two-input node)

2-입력 노드는같은 규칙의 다른 조건절 트리플에 있는 동일한 이름의 변수값과 비교(Match)하는 조건을 나타내는 노드이다. 예를 들어, (?x P ?y) (?y P ?z)라는 두 개의 트리플로 이루어진 조건절에서는 변수 ?y에 대한 2-입력 노드가 만들어지게 된다.

3.1.3 내장함수 노드(Built-in node)

내장함수 노드는 SWRL의 내장함수 조건절을 표현하는 노드로서, 조건절에서의 위치에 상관없이 네트워크 상에서 말단노드 바로 앞에 위치하여 함수 값의 참/거짓 값을 판정하게 된다. 예를 들어 조건절이 "(?x P Q) (+ ?y ?x 3) (?y S T)"일 경우 (?x P Q)와 (?y S T)에 대한 입력 노드들이 생성된 후 내장함수 (+ ?y ?x 3)를 해석하여 "?x + 3의 값이 변수 ?y의 값과 같은지"에 대한 판정을 하게 된다.

3.1.4 말단 노드(Terminal node)

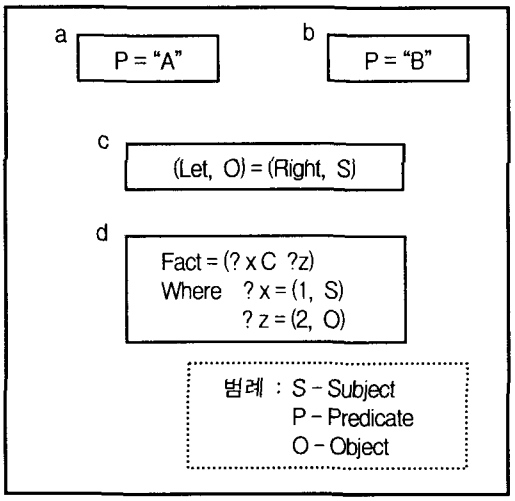
말단 노드는 네트워크의 상위 노드들(1-입력, 2-입력, 내장함수 노드 등)을 거쳐온 결과의 값을 이용하여 결론절의 트리플을 사실베이스에 추가한다. 결론절에 변수가 있을 경우 조건절에서 바인드 된 값을 이용하여 상수로 바꾼 후 사실베이스에 추가한다.

간단한 예제를 가지고 부연 설명하도록 하겠다. 예를 들어 다음과 같은 규칙 베이스와 사실 베이스가 있다고 하자.

<표 1> RDF 표현 예제

규칙 베이스	R1) If (?x A ?y) and (?y B ?z) Then (?x C ?z)
사실 베이스	F1) (1 A 2) F2) (2 B 3)

먼저 규칙 R1의 (?x A ?y)에서 상수 "A"에 대한 상수형 1-입력 노드가 1개 만들어진다. 주의할 점은, 변수 ?x와 ?y는 동일한 이름이 아니므로 변수형 1-입력 노드와는 상관이 없으며, 따라서 네트워크 상 노드가 만들어지지는 않는다. 다음으로 (?y B ?z)에서 상수 "B"에 대한 상수형 1-입력 노드가 만들어진다. 또한 변수 ?y가 첫 번째 조건절 트리플 (?x A ?y)에 나타나므로 2-입력 노드가 만들어진다. 그리고, 마지막으로 결론절에 대한 말단 노드가 만들어진다. 이와 같이 만들어진 네트워크는 다음 그림과 같다.



[그림 1] 규칙 네트워크 예제

위 그림에서 노드 a와 b는 상수형 1-입력 노드들로서 트리플의 Predicate이 "A" 또는 "B"일 때 참이라는 조건을 나타내고 있다. 노드 c는 왼쪽 노드(= 노드 a)에서 참인 트리플의 Object가 오른쪽 노드(= 노드 b)에서 참인 트리플의 Subject와 같을 때 참이라는 조건을 나타내고 있다. 노드 c의 조건을 만족하는 두 개의 트리플 쌍은 말단노드인 노드 d에서 활용된다. 노드 d는 노드 c에서 온 두 개의 트리플들 중 1번째 트리플의 Subject을 값을 변수 ?x의 값으로 하고, 2번째 트리플의 Object의 값을 변수 ?z의 값으로 하여, (?x C ?z)에 바인드함으로써 새로운 사실을 만들어 상충집합과 사실베이스에 추가한다. 구체적인 내용은 다음 절에서 설명된다.

3.2 상충집합 생성을 위한 초기 인터프리트

상충집합은 규칙과 그 규칙에 의해 새로 알려진 사실의 쌍(<규칙, 사실>)의 집합이다. 상충집합은 컴파일된 규칙 베이스(네트워크)에 사실베이스를 적용(Step 2)함으로써 초기화되고, 상충집합의 한 요소를 꺼내어 컴파일된 규칙 베이스에 적용(Step 4)할 때마다 갱신된다. 본 절에서는 상충집합의 초기화에 대하여 앞의 예제를 가지고 설명하도록 하겠다.

상충집합 초기화를 위하여 사실베이스의 모든 사실들이 각각 네트워크에 적용된다. 우리의 예제에서 먼저 사실 (1 A 2)가 네트워크에 적용되는 것을 보자. 먼저 노드 a에 적용되면 Predicate이 "A"이므로 노드 a가 만족된다. 따라서, 노드 c로 가게된다. 노드 c에서는 비교할 오른쪽(Right) 트리플이 없으므로 일단 노드 c의 왼쪽(Left) 버퍼에 저장된다. 그리고, 아직 노드 c를 만족한 상태가 아니므로 노드 d로는 가지 않는다. 또, 네트워크의 모든 루트(Root) 노드와 비교하여야 하므로 노드 b에 (1 A 2)를 또 적용하게 되는데, Predicate이 "B"가 아니므로 만족되지 않는다. 따라서, 노드 c로 내려가지 못한다.

다음으로 (2 B 3)가 네트워크에 적용된다. 먼저 노드 a에 적용하면, Predicate이 "A"가 아니므로 만족되지 않는다. 다음에, 노드 b에 적용하면 Predicate이 "B"이므로 참이 되고, 따라서 노드 c로 내려간다. 노드 c에는 왼쪽 버퍼에 (1 A 2)가 저장되어

있으며, (1 A 2)의 Object와 (2 B 3)의 Subject가 일치하므로 참이 되어 노드 d로 내려간다. 노드 d는 "(1 A 2), (2 B 3)"의 두 개의 트리플 쌍을 받아들여서 1번째 트리플의 Subject 1을 변수 ?x에 바인드하고, 2번째 트리플의 Object 3를 변수 ?z에 바인드함으로써 (1 C 3)이라는 새로운 사실을 사실베이스에 추가함과 동시에 <R1, (1 C 3)> 쌍을 상충집합에 추가한다.

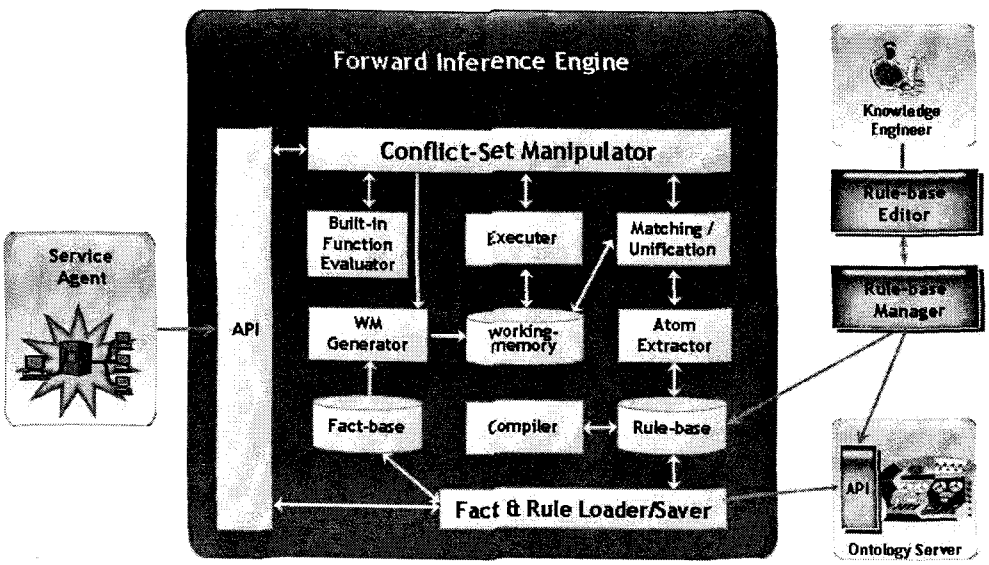
본 예제의 초기화 결과 <R1, (1 C 3)>이 상충집합에 추가되었고, (1 C 3)이 사실베이스에 추가되었다.

3.3 인터프리트

상충집합의 한 요소를 꺼내어 컴파일된 규칙베이스에 적용하는 것을 인터프리트한다고 한다. 인터프리트는 앞 3.2절 초기화에서 설명한 것과 동일하며, 다만 사실베이스의 모든 사실들에 대해서가 아니라, 상충집합의 한 요소에 대해서만 규칙베이스(네트워크)를 적용한다는 점만 다르다.

앞의 예제를 계속하면, 초기화 결과 <R1, (1 C 3)>이 상충집합에 있으므로, 이것을 꺼내어 사실 (1 C 3)을 네트워크에 적용(인터프리트)하게 된다. 그러나, 노드 a와 노드 b를 다 만족하지 못하므로 새로운 상충집합 요소는 생성되지 않는다. 그리고, <R1, (1 C 3)>을 상충집합에서 꺼내면서 제거하였으므로 상충집합은 공집합이 되었다. 그리하여, (Step 3)의 조건에 따라 정방향 추론이 끝나게 된다. 결론적으로, 우리의 예제에서 (1 C 3)이라는 새로운 사실을 도출하고 정방향 추론이 성공적으로 끝나게 된다.

지금까지 설명한 Rete 알고리즘의 장점은 매회(Iteration)마다 사실베이스의 모든 사실들과 규칙베이스의 모든 규칙들의 조건절을 반복하여 비교(Match)하는 대신에, 참으로 판명된 사실들을 2-입력노드의 왼쪽 또는 오른쪽 버퍼에 저장함으로써 다음 회(Iteration)에 중복하여 비교하는 부담



[그림 2] 시맨택 웹 기반 정방향 추론엔진의 시스템 구조도

을 줄여서 추론 속도를 극대화하는데 있다.

그러나 기존의 Rete 알고리즘은 OPS5라고 하는 방법을 사용하고 있어 우리가 개발하려고 하는 SMART-F에서 사용하는 트리플 구조에 사용하기가 어렵다[8]. 따라서 본 연구에서는 기존의 Rete 알고리즘을 트리플 및 내장함수로 표현된 조건절과 결론절에 적용할 수 있도록 수정, 고안함으로써 시맨틱 웹 환경에서 정방향 추론에 적용하기 어려웠던 문제점을 해결 하도록 하였다.

4. SMART-F의 개발

4.1 시맨틱 웹 기반 추론엔진의 요구기능 분석

본 연구에서는 차세대 시맨틱 웹 환경에 알맞은 효율적인 정방향 추론엔진을 설계하고 개발하기 위해 먼저 추론엔진의 요구 기능을 분석하였다.

첫 번째로 차세대 시맨틱 웹 환경에서 소프트웨어 에이전트의 하나의 컴포넌트로서 기능을 지원할 추론엔진은 시맨틱 웹의 표준을 이해하고 처리할 수 있어야 한다. 즉, 시맨틱 웹의 표준인 OWL과 SWRL로 표현된 사실베이스와 규칙베이스를 입력 받아 본 연구에서 설계한 추론 알고리즘에 적용할 수 있는 형식으로 변환이 가능해야 한다는 것이다.

두 번째로 추론엔진에 입력된 정보를 이용하여 도출된 결과가 신뢰할 수 있는 정보여야 한다. 명백(Sound)하고 완전(Complete)한 결과를 도출해 내기 위해서는 입력된 사실과 규칙에 대한 정확한 규칙 네트워크의 구성과 새로운 사실의 생성으로 인한 사실베이스의 갱신 기능이 필요하다.

마지막으로 위에서 언급한 바와 같이, 추론엔진이 에이전트의 하나의 컴포넌트로서 기능을 수행하므로 타 컴포넌트와의 통합 및 인터페이스가 용이해야 한다. 타 컴포넌트로부터 사실베이스와 규칙베이스를 입력 받고, 추론 요청을 받아 추론 과정을 거친 후요청한 컴포넌트에 추론 결과를 반환하는 것이 추론엔진의 프로세스이기 때문에 타 컴포넌트와의 인터페이스는 추론엔진의 중요한 기능 중 하나이다.

이와 같이 분석된 추론엔진의 요구 기능은 추론엔진의 핵심적인 구성요소를 파악하고 전체적인 시스템 구조를 설계하는 기틀을 마련하였다.

4.2 정방향 추론엔진의 구성요소

3절에서 제시한 수정 Rete 알고리즘과 4.1절에서 분석된 추론엔진의 요구 기능에 따라 정방향 추론 엔진은 5개의 핵심 구성요소를 필요로 한다. 구체적인 기능과 역할은 다음과 같다.

4.2.1 사실베이스와 규칙 베이스 형식 변환 및 관리

차세대 시맨틱 웹의 표준인 OWL과 SWRL로 표현된 사실베이스와 규칙 베이스를 적재하여 처리하는 경우 추론엔진에서 사용할 수 있는 형태로 변환하여 저장하고 관리하는 기능을 수행하는 구성요소이다. 이것은 각 표준들로 작성된 문서를 이해하고 추론엔진의 내부적인 형식에 맞게 저장되는 것과 적시에 사용할 수 있도록 관리되는 것을 의미한다.

4.2.2 정확한 규칙 네트워크의 생성 및 관리

정방향 추론엔진의 핵심 구성 요소 중 하나로 추론 알고리즘을 수행하는 초기 단계에 실행된다. 이 구성 요소의 역할은 규칙 베이스의 모든 규칙

들을 컴파일하여 정확한 네트워크의 형태로 구성하는 것이다. 규칙 네트워크는 4가지 종류의 노드로 이루어지는데, 규칙의 조건절과 결론절을 이루는 각 트리플 마다 그 성격에 맞는 형태의 노드를 생성하여 구성한다. 규칙의 결론절을 처리하는 노드에서는 새로 생성된 사실을 상충 집합과 사실베이스에 추가할 수 있는 형태로 저장한다.

4.2.3 규칙 네트워크에 대한 사실의 적용

이 구성 요소는 사실베이스의 사실과 상충집합의 요소 등을 규칙 네트워크에 적용하여 비교하는 기능을 수행한다. 규칙 네트워크의 각 노드를 거치면서 해당 노드에 설정된 값을 사실과 비교하여 조건이 만족할 경우 저장 또는 연결된 다음 노드로 이동하는 역할을 한다. 네트워크의 최종 노드에서는 새로 생성된 사실을 사실베이스와 상충집합에 추가한다.

4.2.4 사실베이스 및 상충 집합의 관리 및 갱신

이 구성 요소는 추론 과정 중에 수시로 사용되는 상충 집합을 관리하는 역할을 한다. 규칙 네트워크가 완성되면 그 규칙 네트워크에 사실베이스의 사실들을 적용하도록 하여 상충집합을 초기화하고 상충집합의 요소를 하나씩 꺼내어 규칙 네트워크에 적용하도록 하는 기능을 수행한다. 네트워크 적용 과정에서 새로운 사실이 추가될 경우 상충집합 또한 갱신된다.

4.2.5 타 컴포넌트와의 통합 및 인터페이스

정방향 추론엔진은 소프트웨어 에이전트의 하나의 컴포넌트로서 존재하기 때문에 에이전트의 요청에 따라 추론 기능을 수행하고, 이 과정에서 도출된 결과를 다시 에이전트에 반환하는 기능을 해야 한다. 이 구성 요소의 역할은 에이전트의 각 컴포넌트와의 인터페이스를 제공하며, 이러한 역할을 하는 구성요소이다.

4.3 시스템 구조

본 연구에서는 Rete 알고리즘을 이용한 정방향 추론엔진과 요구 기능에 따라 <그림 2>와 같은 시스템 구조를 갖는 시맨틱 웹 기반 정방향 추론엔진을 설계하였다. 각각의 구성요소에 대한 기능 및 역할은 다음과 같다.

4.3.1 Fact & Rule Loader / Saver

사실베이스와 규칙 베이스 저장 및 관리를 담당하는 것으로 추론엔진이 실행되면 사실 베이스로부터 OWL로 작성된 사실들과 규칙 베이스로부터 SWRL로 작성된 규칙들을 로드(load)하고 저장하는 역할을 수행한다.

4.3.2 Compiler

규칙 네트워크의 생성 및 관리를 담당하는 것으로 추론엔진 실행 시 규칙 로딩이 완료되면 자동으로 실행된다.

4.3.3 working-memory

Fact & Rule Loader/Saver로부터 로딩된 사실들과 추론 수행 후 생성되는 새로운 사실들을 작업 메모리 요소(WME : working-memory element)로서 저장하는 기능을 수행한다.

4.3.4 Built-in Function Evaluator

규칙 네트워크를 구성하는 노드 중 내장함수 노드의 사용 시 필요하다. SWRL의 내장함수 조

건절을 처리하고 그 결과를 사실에 바인딩(binding)하는 기능을 수행한다.

4.3.5 Executer

규칙 네트워크에 대한 사실의 적용을 담당하는 역할을 수행하며 Conflict-set Manioulator에 의해 호출된다. 하나의 사실을 규칙 네트워크에 적용시켜 추론 과정이 시작되도록 하며, 새로 생성된 사실들을 상충집합과 사실베이스에 추가한다.

4.3.6 Matching/Unification

규칙 네트워크에 대한 사실적용의 일부분인 비교작업을 수행한다. 단일화(unification) 과정은 노드 내에서 바인딩(binding)된 값을 변수에 할당하는 작업을 의미하며, 비교(matching)는 규칙의 트리플과 사실을 비교하는 작업을 말한다. 단일화 과정을 거친 후 비교 작업이 수행된다.

4.3.7 Conflict-Set Manipulator

상충집합의 관리 및 갱신을 담당하는 것으로 각 모듈들을 호출하고 상충집합의 요소들을 관리하며 갱신하는 역할을 한다. 에이전트와 연결되어 추론 결과를 출력하는 기능도 포함한다.

4.4 개발 환경 및 개발 현황

이상과 같이 설계된 정방향 추론엔진 SMART-F는 현재 JDK Ver.1.5를 기반으로 하여 NetBeans Ver.5.0 환경에서 개발을 완료한 상태이다. SWRL로 표현된 규칙 베이스에 대한 저장 및 관리 기능은 이미 개발된 역방향 추론엔진 SMART-B의 모듈을 사용하였으며, RDF 및 OWL로 표현된 사실베이스에 대한 저장 및 관리 기능은 HP사의 Jena Ver.2.3을 이용하였다. 개발된 추론엔진은 시맨틱 웹 기반 정방향 추론의 요구 기능을 충족시켜주고 있다.

5. SMART-F 성능 비교 테스트

위와 같이 개발된 정방향 추론엔진 SMART-F의 성능을 평가하기 위해서 다른 기존의 추론엔진들인 Jena, JESS, Bossam과 추론 속도 비교 테스트를 시행 하였다.

5.1 테스트 온톨로지 및 Rule

추론엔진간의 추론 속도 테스트를 위해 다음 두 가지의 온톨로지와 Rule을 사용하였다.

	Ontology	Rule
Transitive	transitive.owl(1KB)	transitive.swrl
Family	family.owl(21 KB)	hasAunt.swrl

Transitive 온톨로지는 간단히 (a P b), (b P c), (c P d), (d P e)의 4개의 트리플을 가지고 있는 온톨로지이고, Transitive Rule은 이들간의 Transitive 관계를 추론해 주는 Rule로 "(?x P ?y)이고 (?y P ?z)이면 (?x P ?z)이다" 라는 Rule이다.

Family 온톨로지는 직접적인 여러 가지 가족 관계들 즉, 부모, 자식, 형제 등을 표현하고 남자, 여자 등의 간단한 분류를 온톨로지를 통하여 만들어 놓은 것으로 Transitive 온톨로지에 비해 트리플 수가 훨씬 많다. hasAunt Rule은 온톨로지 상의 가족 관계에는 직접적으로 표현되어 있지 않은 Aunt관계를 '부모의 부모의 자식중 여자'라는 관계를 통해서 추론해 내는 Rule이다.

5.2 테스트 결과 및 분석

위에서 설명한 두 가지의 온톨로지와 Rule을 이용하여 SMART-F를 포함한 각각 추론 엔진으로 추론을 해 보았다.

각각의 온톨로지는 각각 3회씩 실험을 시행하였고, 각 실험당 반복횟수는 1000회로 하였다. 테스트 결과는 다음과 같다.

사용 온톨로지	결과 구분	SMART-F	Jena	JESS	Bossam
Transitive (추론 건수 : 6)	건당 추론시간	0.01msec	0.07msec	2.61msec	0.62msec
	초당 추론횟수	100,000회	14,286회	383회	1,613회
Family (추론 건수 : 10)	건당 추론시간	0.043msec	0.108msec	10.43msec	2.322msec
	초당 추론횟수	23256회	9259회	96회	431회

결과에서 보듯이, SMART F는 Transitive에서는 건당 추론 시간이 다른 추론엔진 중 가장 빠른 Jena의 0.07msec보다도 빠른 0.01msec의 속도를 보였고, Family에서도 Jena의 0.108msec보다 빠른 0.043msec의 속도를 보여주었다. 이는 초당 추론 횟수를 비교해 보아도 현저하게 차이가 난다는 것을 확인 할 수 있다.

이로써 다른 알고리즘을 사용하는 엔진이나, 같은 Rete 알고리즘을 사용하고 있는 Jena보다도 개발한 SMART-F가 더 훌륭한 성능을 보여 준다는 것을 증명할 수 있다.

6. 결론

시맨틱 웹과 웹 서비스(Web Services) 기술을 기반으로 시맨틱 웹 서비스로 이어지는 최근의 웹 기반 컴퓨팅 환경의 변화는 머지않은 미래에 컴퓨터간의 상호 운영성의 수준을 한 단계 높은 수준, 즉 컴퓨터간에 서로 이해하고 서로 운용할 수 있는 수준으로 비약시킬 것으로 예상된다. 그러나 이러한 컴퓨팅 환경의 변화는 사용자에게 새롭고 편리한 수단을 제공하는 반면에, 다른 한편으로는 도전할만한 해결해야할 과제들를 던져주고 있다. 컴퓨터 간에 교신된 데이터를 상호이해하고 운용할 수 있다 하더라도 이를 바탕으로 다른 컴퓨터나 인터넷 서비스(웹 서비스 등)와 결합하여 주어진 과제를 수행할 수 있는 방법론이 제공되지 않는 한 컴퓨터간의 상호운영성은 그 한계를 드러낼 수 밖에 없는 것이다.

이러한 과제에 대한 시맨틱 웹 분야에서의 해결책은 규칙 혹은 논리 프레임워크를 이용하여 데이터에 대한 상호운영성의 효과를 극대화하고 있다. 이에 대한 현재 진행 중인 표준이 바로 RuleML을 기반으로 한 SWRL(Semantic Web Rule Language)로서 웹에서의 표준 규칙 언어이다. 본 연구에서는 이와 같이 추진 중인 SWRL의 표현력과 표준을 완벽하게 활용할 수 있는 추론 엔진 개발 연구 일부분의 성과로써 Rete 알고리즘을 이용한 정방향 추론엔진인 SMART-F(SeMantic web Agent Reasoning Tools - Forward chaining inference engine)의 구조적 특성과 관련 기술 및 표준을 소개하고, Java 프로그래밍 언어를 이용하여 단위 컴포넌트로 개발하였다. SMART 프로젝트는 이미 개발된 역방향 추론엔진인 SMART B[2]와 더불어 완성된 형태의 시맨틱 웹 추론 엔진을 지향하고 있다. 또한, SMART 프로젝트의 양대 추론엔진인 SMART-F와 SMART-B는 유비쿼터스 환경에서의 활용성 측면에서 뿐만 아니라, 사실 베이스 및 규칙 베이스의 표현 언어와 관리도구 등을 완벽하게 호환가능함으로써 향후 시맨틱 웹에서의 지식활용을 극대화하여, 차세대 시맨틱 웹 응용 소프트웨어 에이전트의 핵심 도구로 자리잡게 될 것으로 기대하고 있다.

참고문헌

[1] 김홍기, 웰드와이드웹에서 시맨틱 웹으로, *마이크로소프트웨어*, 2002.

[2] 송용욱, 홍준석, 김우주, 윤숙희, 이성규, "차세대 웹을 위한 SWRL 기반 역방향 추론엔진 SMART B의 개발", 한국지능정보시스템학회논문지, 12권 2호(2006), 67~81.

[3] Berners-Lee, T., The Semantic Web, *Scientific American*, Vol.501(2001).

[4] Blaze Advisor, http://www.blazesoft.com/.

[5] Expertise2go, http://www.expertise2go.com/.

[6] EXSYS, http://www.exsys.com/.

[7] EXSYS, Inc., Moving an EXSYS Application to the EXSYS Web Runtime Engine(WREN).

[8] Forgy, C. L, "RETE-A fast algorithm for the many pattern/many object pattern matching problem", *Artificial Intelligence*, Vol.19 (1982), 17~37.

[9] Giarratano, Joseph and Gary Riley, *Expert Systems : Principles and Programming*, 2nd Edition, PWS Publishing Company, 1994.

[10] ILog, http://www.ilog.com/.

[11] RDF Primer, http://www.w3.org/TR/rdf primer/.

[12] Russell, S. and P. Norvig, *Artificial Intelligence - A Modern Approach*, Prentice Hall, 1995.

[13] W3C, http://www.w3.org/Submission2004/ SUBM-SWRL-20040521/.

Abstract

Development of Forward chaining inference engine SMART-F using Rete Algorithm in the Semantic Web

Kyunbeom Jeong* · Yong Uk Song* · June Seok Hong** · Wooju Kim* · Myung Jin Lee* · Ji Hyoung Park*

Inference engine that performs the brain of software agent in next generation's web with various standards based on standard language of the web, XML has to understand SWRL (Semantic Web Rule Language) that is a language to express the rule in the Semantic Web.

In this research, we want to develop a forward inference engine, SMART-F (SeMantic web Agent Reasoning Tools-Forward chaining inference engine) that uses SWRL as a rule express method, and OWL as a fact express method.

In the traditional inference field, the Rete algorithm that improves effectiveness of forward rule inference by converting if-then rules to network structure is often used for forward inference. To apply this to the Semantic Web, we analyze the required functions for the SWRL-based forward inference, and design the forward inference algorithm that reflects required functions of next generation's Semantic Web deducted by Rete algorithm. And then, to secure each platform's independence and portability in the ubiquitous environment and overcome the gap of performance, we developed management tool of fact and rule base and forward inference engine. This is compatible with fact and rule base of SMART-B that was developed. So, this maximizes a practical use of knowledge in the next generation's Web environment.

Key Words : Forward Chaining Inference Engine, OWL, RDF, Semantic Web, SWRL, XML

* Dept. of Management Information Systems, Yonsei University
** Department of Management Information Systems, Kyonggi University