온톨로지 디버깅을 위한 MEXS 추출 및 저장 기법
(MEXS Extracting and Storing for Ontology Debugging)

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

요 약 현재 온톨로지를 구축하는데 있어 OWL을 온톨로지 표현 언어로 많이 사용하는 추세이다. OWL 온톨로지의 내제된 정보(클래스간의 계층구조, 인스턴스의 정확한 타입)를 추론하기 위해, 현재 많은 온톨로지 추론엔진이 개발되어지고 있다. 그러나 대부분의 온톨로지 추론 엔진들은 단순히 추론 결과만 명시할 뿐, 그 과정을 표현하지는 않는다. 따라서 본 논문에서는 논리적으로 정당하지 못한 온톨로지를 디버깅 하기위한 MEXS(Minimum Expression Axiom Set) 추출과 저장에 대한 기법을 제안한다. MEXS를 추출하기 위해서는 온톨로지 내에서 논리적인 오류를 유발하는 Axiom들을 찾아내는 방법은 매우 중요하다고 할 수 있다. 이에 본 논문은 두 가지 부분에 초점을 맞추어 연구를 진행하였다. 첫 번째, 논리적으로 정당하지 못한 온톨로지가 주어졌을 때, 논리적 오류를 유발하는 핵심 Axiom을 찾아내고, 이와 연관이 되는 Axiom들을 찾아낸다. 두 번째, 논리적으로 정당하지 못한 온톨로지를 디버깅하기 위한 MEXS를 구성한다. 본 연구 결과는 서술 논리에 기반을 둔 모든 어플리케이션에 적용이 가능하다.

키워드 : 온톨로지, 서술 논리, 온톨로지 추론, 온톨로지 디버깅

Abstract The web ontology language (OWL) has been used by ontology designers to construct ontology. In order to derive hidden information(concept subsumption, concept satisfiability and realization) of OWL ontology, a number of OWL reasoners have been introduced. But most reasoners simply report these information without process for any arbitrary entailment and unsatisfiable concept derived from a OWL ontologies. In this paper, we propose Minimum Expression Axiom Set (MEXS) detection and storing for debugging unsatisfiable concepts in ontology. In order to detect MEXS, we need to find axiom to cause inconsistency in ontology. Therefore, our work focused on two key aspects: given a inconsistency ontology, identifying the roots of axioms to occur unsatisfiable and derived axioms from among them; and extracting MEXS. Our results can be applicable to all application, which is at the basis of the description logic.

Key words : Ontology, Description Logic, Ontology Reasoning, Ontology Debugging

1. 서 론

시맨틱 웹은 현재의 웹을 확장하여 웹에 올라오는 정보에 정형화된 의미를 부여함으로써 컴퓨터와 사람이 협동적으로 작업을 수행할 수 있도록 프레임워크를 제공한다[1]. 이러한 시맨틱 웹의 중추적인 역할을 담당하는 온톨로지는 특정 개념에 대해 공유가 가능하도록 정형화된 형식으로 명시한 명세 사항이다[2]. 즉, 어느 특정 도메인에 관련된 단어들을 계층적으로 표현하고, 추가적으로 이를 확장할 수 있는 추론규칙(Inference Rule)을 포함한다[3]. 온톨로지의 중요한 특징은 같은 개념에 대해서 서로 다른 단어를 사용하더라도 이를 해결할 수 있으며, 각각의 개념에 한정 조건(Restriction)을 명시함으로써 개념들 간의 계층 구조를 추론할 수 있다.

현재 온톨로지를 구축하는데 있어 OWL[4]을 온톨로지 표현 언어로 많이 사용하는 추세이다. OWL은 여러 가지 형태가 존재하지만 대부분이 서술논리(Description Logic)에 근거하여 표현되고 있다. 따라서 온톨로지 추론은 서술논리 추론과 매우 밀접한 관계를 맺고 있다.

· 본 논문은 숭실대학교의 지원을 받았습니다.
· 이 논문은 2007 한국컴퓨터종합학술대회에서 '온톨로지 디버깅을 위한 MEXS 추출 및 저장 기법'의 제목으로 발표된 논문을 확장한 것임

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

논문접수 : 2007년 10월 1일
심사완료 : 2008년 4월 22일

정보과학회논문지: 소프트웨어 및 응용 제35권 제6호(2008.6)

점차 온톨로지의 규모가 대용량화 되고, 하나의 온톨로지 구축에 참여하는 사람이 많아지면서 온톨로지 내부에서 논리적인 오류가 많이 발생한다. 이러한 논리적인 오류를 찾아내기 위해서, 서술논리 기반의 온톨로지 추론엔진이 필요하다. OWL 온톨로지의 내재된 정보(클래스간의 계층 구조, 인스턴스의 정확한 타입)를 추론하기 위해, 현재 많은 온톨로지 추론엔진이 개발되어지고 있다. 그러나 대부분의 온톨로지는 단순히 추론 결과만을 명시할 뿐, 그 과정을 표현해 주지는 않는다. 대부분의 온톨로지 디자이너는 서술 논리에 대해 지식이 충분치 않으며, 서술 논리의 전문가라 할지라도 온톨로지의 용량이 크면, 오류를 쉽게 해결할 수는 없다. 따라서 본 논문에서는 논리적으로 정당하지 못한 온톨로지를 디버깅하기 위한 MEXS(Minimum Expression Axiom Set) 추출과 저장에 대한 기법을 제안한다. MEXS는 온톨로지 상에서 논리적인 오류를 유발하는 공리들이나, 클래스간의 계층 정보가 추론 되었을 때, 추론에 바탕이 되는 공리들이 모인 집합이라고 정의 할 수 있다. MEXS를 추출하기 위해서는 온톨로지 내에서 논리적인 오류를 유발하는 공리들을 찾아내는 방법은 매우 중요하다고 할 수 있다. 이에 본 논문은 두 가지 부분에 초점을 맞추어 연구를 진행하였다. 첫 번째, 논리적으로 정당하지 못한 온톨로지가 주어졌을 때, 논리적 오류를 유발하는 핵심 공리를 찾아내고, 이와 연관이 되는 공리들을 찾아낸다. 두 번째, 논리적으로 정당하지 못한 온톨로지를 디버깅하기 위한 MEXS를 구성한다.

본 논문에서 제안하는 방식은 Tableaux 알고리즘 기반의 MEXS 구성 기법과 이를 좀 더 최적화한 Hitting Set Tree 기반의 MEXS 구성 기법이다. Tableaux 알고리즘 기반의 MEXS 구성 기법은 *SHOIN* 수준의 서술 논리 추론이 가능한 확장 Tableaux 알고리즘을 기반으로 하고 있으며, Hitting Set Tree 기반의 MEXS 구성은 특히 대용량 온톨로지의 다수의 논리적인 오류를 일으키는 공리를 효율적으로 찾아내기 위해 Hitting Set Tree를 적용한 방법이다. 현재 제안된 2가지 방법은 glass-box 방식이므로, 추론 엔진 성능에 영향을 받는다. 따라서 제안된 방법을 효율적으로 실용화할 온톨로지 추론엔진과 추론 엔진에 영향을 받지 않는 black-box 기법은 향후에 연구될 예정이다.

2. 서술 논리(Description Logic)와 Tableaux 알고리즘

온톨로지 추론의 기반이 되는 서술 논리(Description Logic)는 기본적으로 Concepts, Roles, Individuals로 사람이 가지고 있는 지식을 표현한다[5]. 따라서 본 논문에서 표현하고자 하는 MEXS를 추출하기 위해서는 서술 논리에서 표현하는 T-Box와 A-Box의 구조를 이해할 필요가 있다. T-Box에는 Terminological 부분이 선언되는데, 이것은 온톨로지의 Class와 Property 및 Restriction 정의 부분이다. A-Box에는 Assertional 부분이 정의되는데, 이것은 온톨로지의 Instance 정의 부분이다. 이처럼 본 논문에 적용된 온톨로지 언어인 OWL과 서술 논리는 그 구조가 일치하기 때문에 쉽게 하나의 시스템으로 연결되어질 수 있다.

서술 논리의 추론은 크게 세 가지로 나뉜다. 첫 번째는 Subsumption Relation을 추론하는 것을 의미하는데, 이 과정에서 온톨로지 클래스들의 Super/SubClass의 관계를 추론하게 된다. 여기서 Subsumption은 하나의 Concept이 다른 Concept을 포함하는 것을 의미한다. 두 번째는 Insanitation 관계를 추론하는 것으로써, 하나의 인스턴스가 어느 Concept에 속하는지 추론한다. 마지막은 온톨로지의 논리적 정당성을 체크하는 것으로서, 클래스의 Restriction 정의 부분에서 논리적 오류가 있을 경우, 논리적 오류를 유발하는 클래스를 검사한다.

일반적으로 Subsumption 관계는 $C \subset D$와 같은 형식으로 표현되는 데, 이를 증명하기 위해서는 $\neg C \cup D$가 satisfiable[6] 하다는 것을 보이면 된다. 이는 $C \subset D$가 satisfiable 하다는 것을 증명하기 위해서는 $\neg C \cup D$의 부정인 $C \cap \neg D$가 만족불능하다는 것을 보이면 된다는 것을 의미하기도 한다.

Tableaux[7] 알고리즘은 satisfiability 테스트를 실험적으로 Tractable하게 보여줄 수 있음을 증명하였고, 현재 많은 서술 논리 기반의 온톨로지 추론 기술은 Tableaux 알고리즘을 기반으로 하고 있다. 따라서 제안하고자 하는 MEXS 추출에 바탕이 되는 온톨로지 추론 방법에는 Tableaux 알고리즘 기반의 서술 논리 추론을 적용되었으며, 이를 기반으로 추론 도중에 해당 온톨로지에 논리적인 오류를 발생하는 공리, 또는 $C \cap \neg D$가 만족불능하는데 원인을 제공하는 공리를 추출한다.

Tableaux 알고리즘의 기본 아이디어는 증명하고자 하는 내용의 부정에 대해서 다양한 변환 규칙을 적용하여 satisfiable하지 않다는 것을 보여주는 방식을 취하고 있다. 서술 논리는 $\cap$, $\cup$, $\neg$, $\exists$, $\forall$등으로 표현되기 때문에 Tableaux 알고리즘에서는 이에 대한 Expansion 규칙을 반복적으로 적용하면서 탐색공간을 확장하게 된다. 그리고 모든 탐색 공간의 말단 노드가 모순(Contradiction)이라는 것을 보여줌으로써 증명하고자 하는 Subsumption 관계의 부정이 만족불능하다는 것을 증명하게 된다. 다음 표 1은 Tableaux 알고리즘에서 활용하는 기본 확장 규칙의 일부를 보여준다.

3. 온톨로지 추론엔진

표 1 Tableaux 알고리즘

∩-rule	if 1. $(C_1 \cap C_2) \in \mathcal{L}(x)$ 2. $\{C_1, C_2\} \not\subseteq \mathcal{L}(x)$ then $\mathcal{L}(x) \rightarrow \mathcal{L}(x) \cup \{C_1, C_2\}$
∪-rule	if 1. $(C_1 \cup C_2) \in \mathcal{L}(x)$ 2. $\{C_1, C_2\} \cap \mathcal{L}(x) = 0$ then a. save T b. try $\mathcal{L}(x) \rightarrow \mathcal{L}(x) \cup \{C_1\}$ If that leads to a clash then restore T and c. try $\mathcal{L}(x) \rightarrow \mathcal{L}(x) \cup \{C_2\}$
∃-rule	if 1. $\exists R.C \in \mathcal{L}(x)$ 2. there is no y s.t $\mathcal{L}(<x,y>)=R$ and $C \in \mathcal{L}(y)$ then create a new node y and edge <x,y> with $\mathcal{L}(y)=\{C\}$ and $\mathcal{L}(<x,y>)=R$
∀-rule	if 1. $\forall R.C \in \mathcal{L}(x)$ 2. there is some y s.t. $\mathcal{L}(<x,y>)=R$ and $C \notin \mathcal{L}(y)$ then $\mathcal{L}(y) \rightarrow \mathcal{L}(y) \cup \{C\}$

온톨로지 추론엔진을 구축하는 연구는 OWL로 표현된 온톨로지에 대해 정확하면서 완전한 추론을 납득할 만한 시간 내에 수행할 수 있는 알고리즘 개발에 집중되고 있다. 물론 정확하거나 완전함보다는 표현력이 높은 온톨로지 표현 언어에 대해서 온톨로지 추론 엔진을 연구하는 패러다임도 있지만, 이와 같은 연구는 추론 엔진의 정확성, 완전성, 시간 복잡성 등을 보장하게 되지 못하게 되므로 활용성이 한정될 수밖에 없다.

온톨로지를 표현하는 기반으로 주로 서술 논리가 사용되어 지고 있으나, 다른 한편에서는 F-논리[8] 기반으로 온톨로지를 표현하는 연구가 매우 활발하게 진행되고 있다. F-논리로 온톨로지를 표현하는 경우는 시맨틱 웹에서 중요하게 고려되고 있는 Property의 계층적 구조에 대한 표현 및 추론에 제한이 있다는 문제점이 있지만, 이러한 제약성이 허용되는 온톨로지 구축 분야에서는 매우 성능이 우수한 온톨로지 추론 엔진이 개발되고 있다.

현재 많이 활용되고 있는 OWL 기반의 온톨로지 추론 엔진은 다음 표 2와 같은데, Ian Horroks에 의해 만들어진 FaCT(Fast Classification of Terminology)[7]를 시작으로 앞장에서 언급한 Tableaux 알고리즘을 효율적으로 구현하고 있다.

표 2에서 언급한 온톨로지 추론엔진은 같은 목적을 가지고 개발되고 있으며, 현재 대용량의 온톨로지 추론을 실시간으로 처리하는 문제를 극복하기 위한 다양한 연구가 활발하게 진행되고 있다. 현재 FaCT, FaCT++는 매우 많이 이용되고 있으며, 최근에는 Pellet과 KAON2의 활용도 점차 증가되고 있는 추세다. 현재 개발된 온톨로지 추론엔진은 온톨로지 저작도구와 연동되어 두 가지 추론 서비스를 시행한다. 첫 번째는 온톨로지를 구성하고 있는 Class들의 논리적인 정당성을 확인해 주는데, 이와 같은 기능은 구축된 온톨로지의 무결성을 확인하는데 유효하게 이용될 수 있다. 두 번째는 Class들 간의 숨겨진 계층 관계를 찾아준다. 그러나 대부분의 온톨로지 추론 엔진들은 단순히 추론 결과만 명시할 뿐, 그 과정을 표현하지는 않는다. 따라서 본 논문에서는 기존의 추론 엔진에 적용된 서술논리를 기반으로 논리적으로 정당하지 못한 온톨로지를 디버깅 하기 위한 MEXS(Minimum Expression Axiom Set) 추출과 저장에 대한 기법을 제안한다.

표 2 온톨로지 추론 엔진

온톨로지 추론 엔진	개발 학교/회사	구현언어
FaCT, FaCT++	Ian Horroks of U. of Manchester	LISP, C++ 기반의 오픈소스
Pellet	U. of Maryland, MIND Lab.	JAVA 기반의 오픈 소스
KAON2	U. of Karlsruhe	JAVA 기반의 오픈 소스
CEL	Dresden Univ.	LISP 기반의 오픈소스
MSPASS	U. of Manchester	C 기반의 오픈소스
RacerPro	Racer/Franz Inc	LISP
Cerebra Engine	Network Inference	C++

4. MEXS(Minimum Expression Axiom Set)

MEXS는 온톨로지 내에서 논리적 오류를 유발하는 Axiom들의 Justification의 집합이다. 논리적 오류가 있는 클래스가 존재할 때, 이 클래스는 Tableaux 알고리즘의 확장 규칙의 반복적인 적용을 통해서 완전 그래프를 그려 나가다가, 결국 Clash를 발생시킨다. Justification은 Tableaux 알고리즘을 적용했을 때의 초기 공리부터 Clash가 발생한 순간까지의 Tableaux 알고리즘 진행에 따른 공리의 상태를 기록하고 있다. 이를 위해 온톨로지에 존재하는 모든 공리에 ID-number를 할당한다. 예를 들어 다음과 같은 공리가 주어졌을 때,

1. $A \subseteq B \wedge \exists R.E$
2. $B \subseteq C \wedge D$
3. $C \subseteq (\geq 1.R) \wedge E$
4. $B \subseteq E$

A는 Tableaux 알고리즘을 적용했을 때의 초기 공리에 해당하므로 [0]이라는 Justification을 갖게 된다. Unfold 규칙과 $\wedge$ 규칙에 따라 그래프 노드에는 B와 $\exists$

R.E가 추가되며, 각각 공리에 해당 ID-number가 추가된다. 이때 현재 공리를 유도하는 이전 공리의 ID-number가 [0]이므로, 이전 ID-number는 삭제되고, 따라서 B와 ∃R.E의 Justification은 [1]이 된다. 다시 Unfold 규칙과 ∧ 규칙에 따라 그래프 노드에는 C와 D가 추가되며, 이때 현재 공리를 유도하는 이전 공리의 ID-number가 [1]이므로, 각 공리에 해당 ID-number가 추가된다. 따라서 C 와 D의 Justification은 [1.2]가 된다. 이러한 방식으로 Justification을 추가해 나감으로써, Clash가 유발되었을 때, 온톨로지 내에서 논리적 오류를 유발하는 골리들을 찾아낼 수 있으며. 이러한 Justification을 기반으로, MEXS를 구성할 수 있게 된다.

MEXS Justification의 정의는 다음과 같다.

- DEFINITION 1 MEXS - 논리적 오류를 유발하는 공리들의 Justification의 집합이며, 항상 최소화한 상태로 유지된다.
- DEFINITION 2 Justification - Tableaux 알고리즘 진행 따라 유도된 공리들의 도출 과정을 기록한 벡터 정보. Bit Set으로 구성된다.

5. Tableaux 알고리즘 기반의 MEXS 구성 기법

본 장에서는 Tableaux 알고리즘을 기반으로 Justification을 기록하고, MEXS를 구성하는 방식에 대해 설명한다. 앞 장에서 기술했듯이 Justification은 Tableaux 알고리즘을 적용했을 때의 초기 공리부터 Clash가 발생한 순간까지의 Tableaux 알고리즘 진행 따른 Axiom의 도출 정보를 기록하고 있다. 따라서 기존의 Tableaux 알고리즘을 그대로 적용하는 것이, Justification을 기록하는데 효율적이다. 즉 Tableaux 알고리즘은 그래프를 구축하면서, Clash가 발생하면 알고리즘 진행을 멈추고, 현재 온톨로지가 논리적 오류가 있음을 메시지로 출력한다. 그러나 Non-deterministic의 규칙이 적용되었을 때는 규칙이 적용된 지점에서 여러 분기(Branch)로 나뉘게 되고, 모든 분기점에서 Clash가 발생되어야 논리적 오류가 존재한다고 결정한다. 따라서 MEXS 안에는 온톨로지 내에서 논리적 오류를 일으키는 모든 공리들의 Justification을 저장하고 있어야 하므로 Clash가 발생하면, 현재 Clash를 유발하는 공리들의 Justification을 MEXS에 저장하고, Clash를 유발하는 그래프 노드의 또 다른 Non-deterministic을 확인하여, 가능한 모든 논리적 오류들을 유발하는 공리들의 Justification을 찾아내야 한다. 따라서 정확한 MEXS를 구성하기 위해서는 기존의 Tableaux 알고리즘의 각각의 규칙들과 종료조건을 수정할 필요가 있다. 이를 위해 각 공리에 Justification과 branching-point의 정보가 함께 저장된다. 그래프 노드를 구성하는 각 공리는 다음과 같이 정의할 수 있다.

Axiom[branching-point set, Justification]

수정된 종료조건은 다음과 같다.

- Tableaux 알고리즘 진행시 모든 Non-deterministic을 고려했으며, 어떠한 Clash도 발생하지 않을 경우, 더 이상 적용할 Tableaux 규칙이 없다면 종료한다.
- Tableaux 알고리즘 진행시 모든 Non-deterministic을 고려하지 않았으며, 어떠한 Clash도 발생하지 않을 경우, 현재 기록된 최근 분기 점(branching-point)으로부터 다시 그래프를 구축한다.
- Tableaux 알고리즘 진행시 모든 Non-deterministic을 고려하지 않았으며, Clash가 발생했을 경우, Clash를 유발한 Justification을 MEXS에 기록하고, 현재 기록된 최근 분기점으로부터 다시 그래프를 구축한다.

다음 그림 1~그림 5는 Tableaux 알고리즘 기반의 MEXS 구성과정을 보여준다. 먼저 온톨로지 추론엔진의 지식베이스에 다음과 같은 공리가 존재한다고, 가정했을 때, 수정된 Tableaux 알고리즘을 진행하면, 그림 1과 같은 결과를 보여준다.

1. A ⊆ B ∧ ∃R.E
2. A ⊆ F ∨ ∀R.B
3. B ⊆ C ∧ D
4. C ⊆ (≥1.R) ∧ E
5. B ⊆ E

E와 -E는 서로 Clash를 일으키며, E는 [1.3.4]라는 Justification을 갖는다. 따라서 E는 1번 공리->3번 공리->4번 공리의 순으로 유도된 공리임을 알 수 있다. 같은 방법으로 -E역시 1번 공리->5번 공리의 순으로 유도되었다. 따라서 Clash를 유발한 두 공리의 Justification을 MEXS에 기록한다. 이때 MEXS는 항상 최소가 되어야 함으로, {1,3,4}∪{1,5}의 결과가 기록된다. 따라서 현재 MEXS K1은 {1,3,4,5}가 된다.

추론엔진의 지식베이스에 다음과 같이 새로운 공리가 추가된다면 그림 2와 3과 같은 결과를 보여준다.

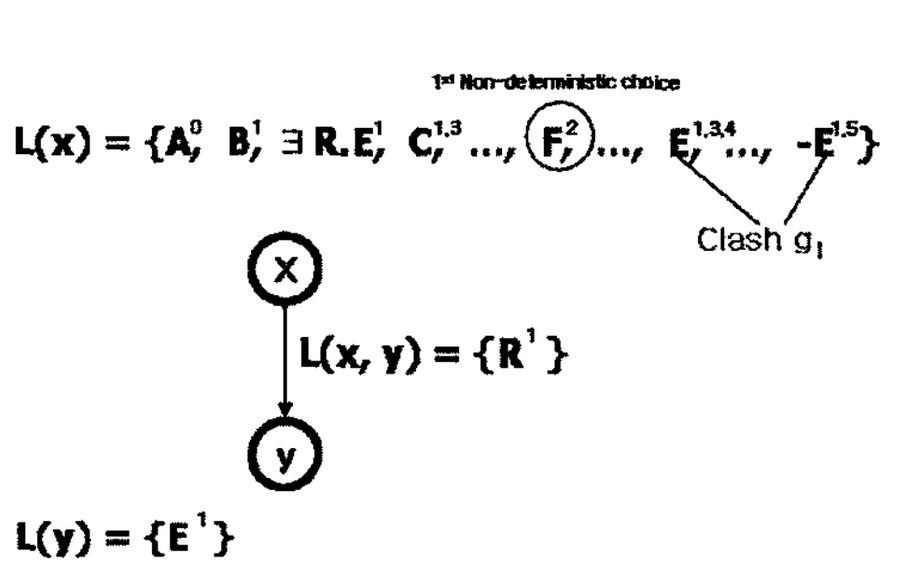

그림 1 MEXS의 구성 과정 1

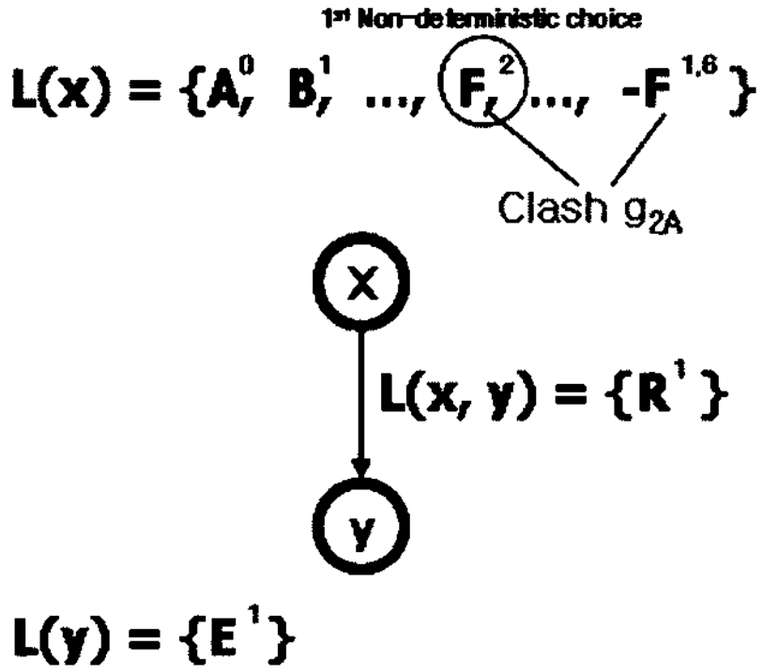

그림 2 MEXS의 구성 과정 2-1

6. B ⊑ -F

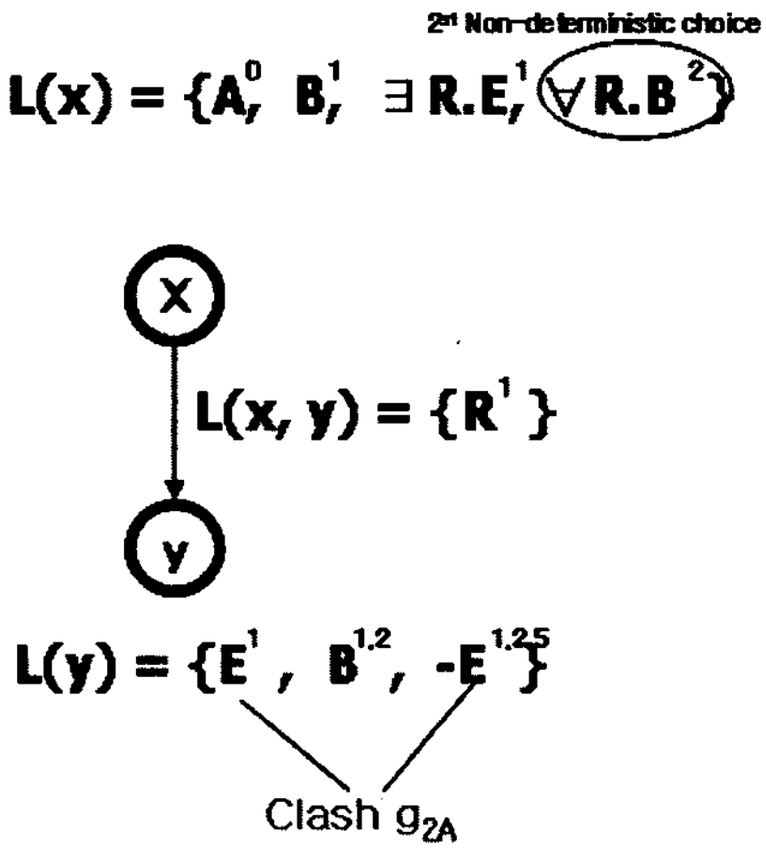

그림 3 MEXS의 구성 과정 2-2

먼저 이전 단계에서 (F ∨ ∀R.B)는 Non-deterministic을 의미한다. 따라서 F가 Non-deterministic으로 선택되었을 경우를 먼저 살펴보면, 2번 공리로부터 유도된 F와 1번 공리->6번 공리로부터 유도된 -F가 Clash를 일으킨다. 따라서 Clash를 유발한 두 공리의 Justification을 MEXS에 기록한다. 다음으로 ∀R.B가 Non-deterministic으로 선택되었을 경우를 고려했을 때, 1번 공리로부터 유도된 E와 1번 공리->2번 공리->5번 공리에서 유도된 -E가 Clash를 일으킨다. 따라서 Clash를 유발한 두 공리의 Justification을 MEXS에 기록한다. 이때 MEXS는 항상 최소가 되어야하므로, {1,2,6}∪{1,2,5}의 결과가 기록된다. 따라서 현재 MEXS K2는 {1,2,5,6}이 된다.

추론엔진의 지식베이스에 다음과 같이 새로운 공리가 추가된다면 그림 4와 5와 같은 결과를 보여준다.

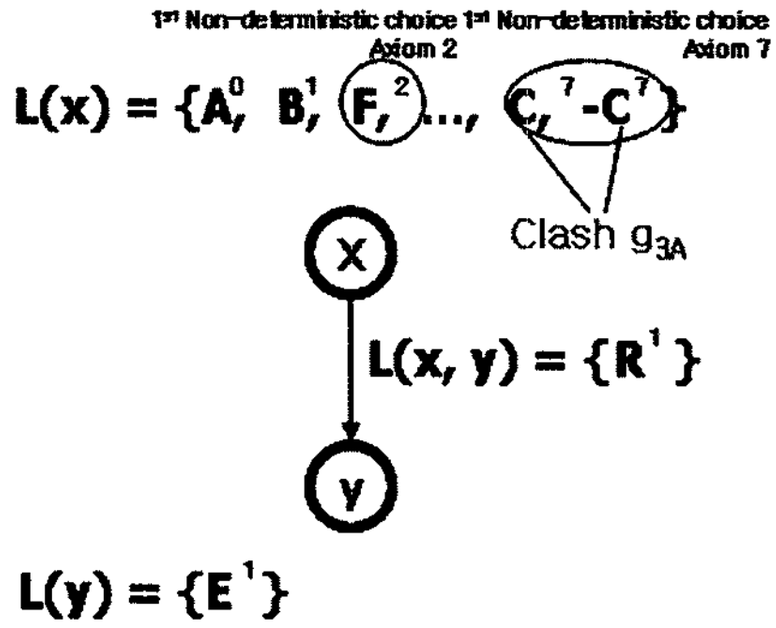

그림 4 MEXS의 구성 과정 3-1

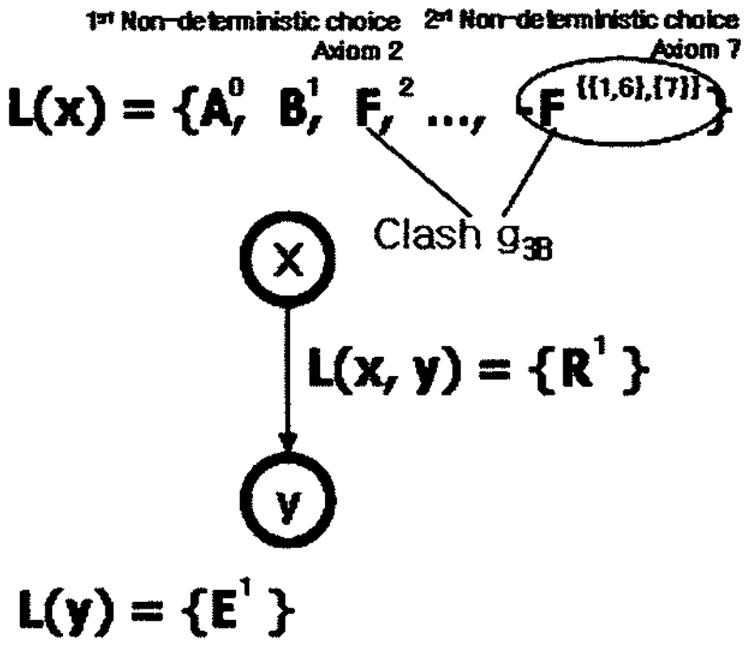

그림 5 MEXS의 구성 과정 3-2

7. A ⊑ (C ∧ -C) ∨ -F

먼저 이전 단계에서(F ∨ ∀R.B)는 첫 번째 Non-deterministic을 의미한다. 또한 새로 추가된 공리의 (C ∧ -C) ∨ -F는 두 번째 Non-deterministic을 의미한다. 따라서 (C ∧ -C)가 Non-deterministic으로 선택된 경우를 먼저 고려했을 때, 이와 같은 경우는 이전 단계의 Non-deterministic의 선택 여부와 상관없이 무조건 Clash를 유발한다. 두 번째로 -F가 Non-deterministic으로 선택된 경우, 이전 단계의 Non-deterministic으로 F가 선택이 되어야 Clash를 유발한다. 따라서 이전 단계의 Non-deterministic 선택에 영향을 받으며, MEXS 구성에 있어서도 현재 선택된 두 개의 Non-deterministic에서의 Justification뿐 아니라, 이전에 선택 되었으며, 현재 Clash상황에 아무런 영향을 미치지 않는 Non-deterministic에 대한 Justification 역시 기록된다. 따라서 현재 MEXS K_2는 {1,2,5,7}이 된다.

6. Hitting Set Tree 기반의 MEXS 구성

Tableaux 알고리즘 기반의 MEXS 구성 기법은 기존의 Tableaux 알고리즘을 그대로 사용할 수 있기 때문

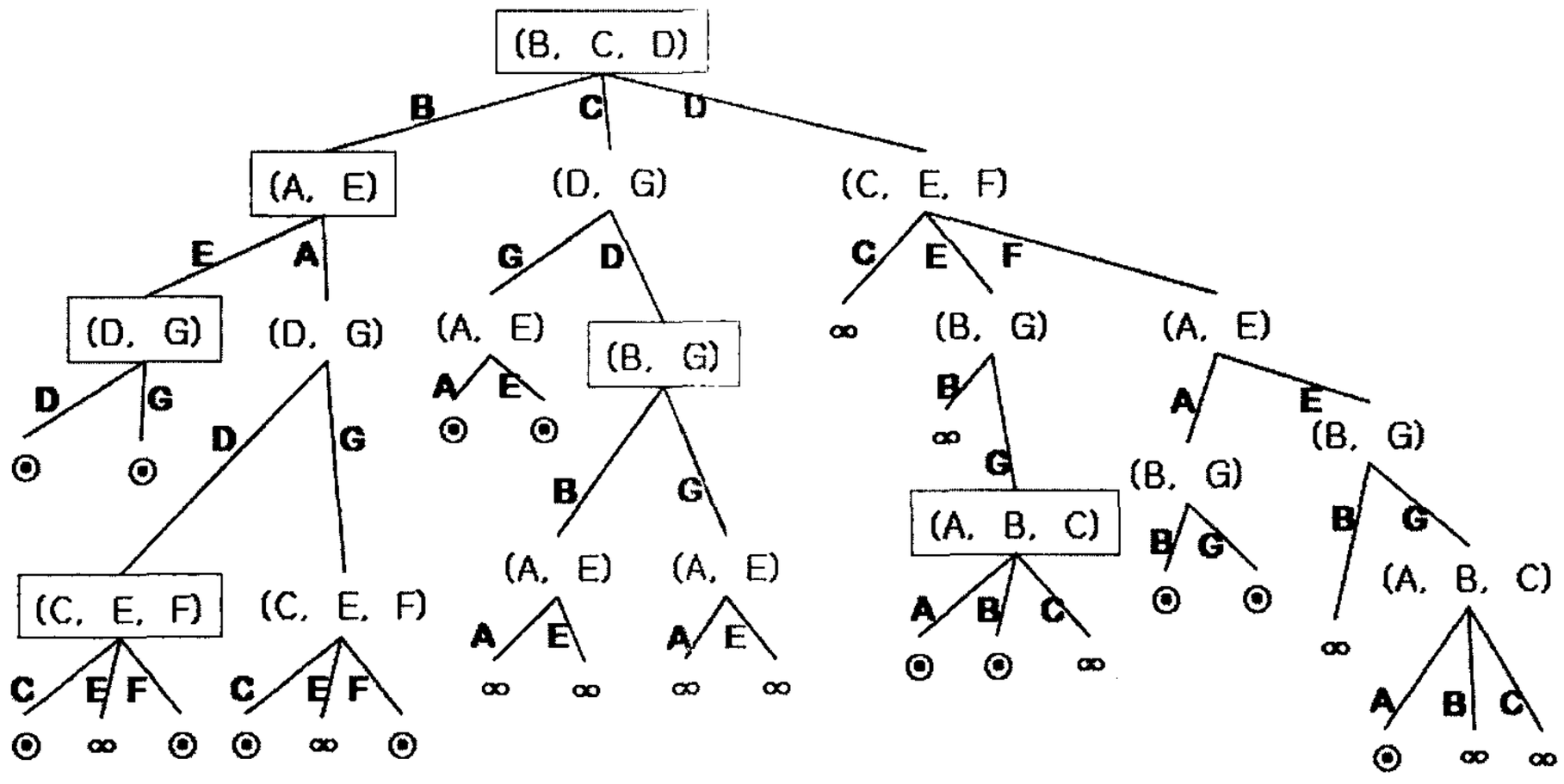

그림 6 Hitting Set Tree를 이용한 MEXS의 구성

에, 대부분의 Tableaux 알고리즘 기반의 온톨로지 추론 엔진에 쉽게 적용이 가능하다는 장점이 있다. 반면에 MEXS를 구성함에 있어서는 모든 Non-deterministic을 고려하기 때문에 탐색 그래프의 모든 거의 모든 노드들을 탐색하게 된다. 따라서 본 장에서는 MEXS를 효율적으로 구성하기 위해 Hitting Set Tree 기반의 MEXS 구성기법을 설명한다.

Hitting Set Tree 기반의 MEXS 구성 기법에서 Justification을 생성하는 방법은 수정된 Tableaux 알고리즘을 그대로 적용한다. 그림 6은 Hitting Set Tree 기반의 MEXS 구성 기법의 예를 보여준다. 일단 Justification이 하나 구해지면(Non-deterministic을 고려하지 않음), Justification을 통해서 임의의 공리를 하나 제거한 후, 다시 온톨로지의 논리적인 정당성을 찾아낸다. 이때 Clash가 발생되면, Clash를 유발하는 Axiom들로 구성된 Justification은 새로운 노드로 생성한다. 즉 초기 Justification은 Hitting Set Tree의 루트 노드가 되고, 임의로 삭제된 공리는 edge가 되며, 새로 구성된 Justification은 하위 노드가 된다. 이러한 방법으로 트리를 구성해 나가다가, 더 이상 Clash를 유발되지 않거나, 이미 이전에 탐색한 적이 있던 경로면 하위노드 생성을 중단하고, 상위 노드로 Back-Tracking 한다. MEXS는 트리가 구축되면서 자연스럽게 생성되는데 이때 이전에 한번 저장되었던 Justification(하위 노드)은 다시 저장 되지 않는다.

7. 실 험

본 논문에서 제안한 Tableaux 알고리즘 기반의 MEXS 구성 기법과 Hitting Set Tree 기반의 MEXS 구성 기법 성능을 비교 분석해 보았다. 성능 비교의 기준은 두 가지로 온톨로지 내에 논리적 오류를 유발하는 클래스의 개수에 따라 MEXS를 구성하는데 걸리는 시간과 MEXS 구성의 정확도이다. MEXS 구성의 정확도는 실제 온톨로지에서 MEXS로 구성된 공리들은 제거한 상태에서 논리적 정당성 테스트와 Subsumption 추론의 정확도를 검사했다.

테스트에 활용된 온톨로지는 PERSON_Relation 온톨로지와 Food 온톨로지이고, PERSON_Relation 온톨로지는 다시 내부의 클래스들에 1개~15개까지 무작위로 논리적 오류를 일으켜 분류하였으며, Food 온톨로지 역시 내부의 클래스들에 1개~10개까지 무작위로 논리적 오류를 일으켜 분류 하였다. 본 실험을 위해 사용한 온톨로지 엔진은 Pellet인데, 현재 Pellet은 오픈 소스이기 때문에 본 실험과 비슷한 유형의 실험에 적용이 쉽다.

그림 7은 PERSON_Relation 온톨로지에 대한 실험 결과를 보여준다. 논리적 오류를 일으키는 클래스의 개수가 8개 이상이 되었을 경우 Tableaux 알고리즘 기반의 MEXS 구성 기법은 기준 시간인 100sec를 넘어간다. 반면 Hitting Set Tree 기반의 MEXS 구성 기법은 보다 안정적인 성능을 보여준다. 그림 8은 Food 온톨로지에 대한 실험 결과를 보여준다. Food 온톨로지는 PERSON_Relation 온톨로지에 비해 Non-deterministic을 고려해야할 공리가 많고, 보다 복잡한 Restriction으로 정의된 클래스가 다수 존재한다. 따라서 추론 시간이 PERSON_Relation 온톨로지 비해 많이 소모된다. 따라서 Tableaux 알고리즘 기반의 MEXS 구성 기법은 기준 시간인 100sec를 넘어간다. 반면 Hitting Set Tree 기반의 MEXS 구성 기법은 보다 적은 시간이 소모 되었다. 그러나 Hitting Set Tree기법 역시 Justification 기록 단계에서 수정된 Tableaux 알고리즘을 사용하므

표 3 MEXS 구성의 정확도 검사

	논리적 정당성 검사		Subsumption 추론	
	MEXS 제거 전	MEXS 제거 후	MEXS 제거 전	MEXS 제거 후
Person_Relation 온톨로지	15 개	0 개	32개	19개
Food 온톨로지	10 개	0 개	57개	21개

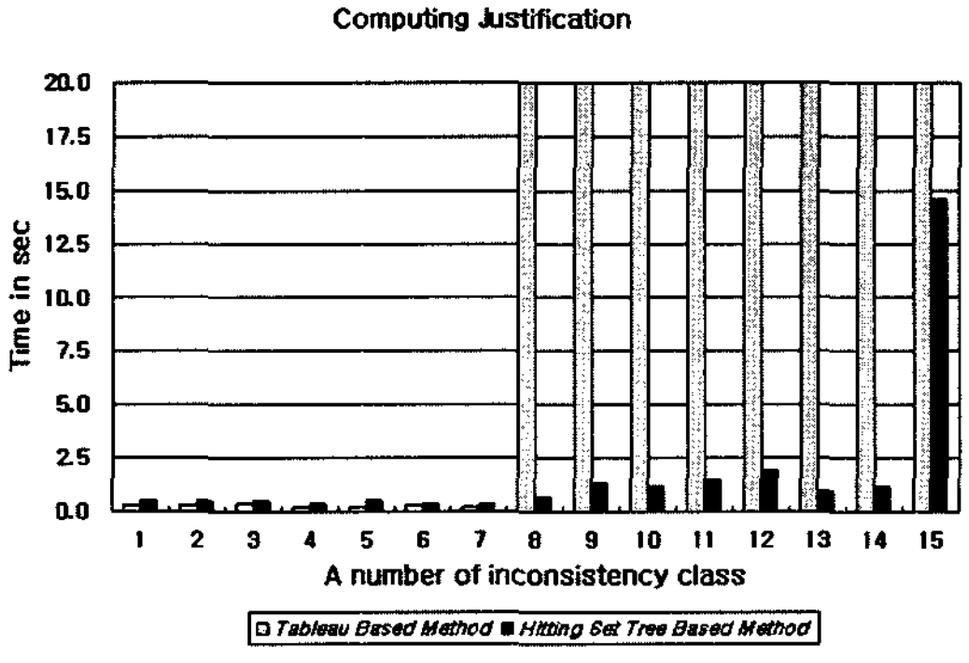

그림 7 PERSON_Relation 온톨로지에서 논리적 오류를 일으키는 클래스 수에 따른 MEXS 구성시간

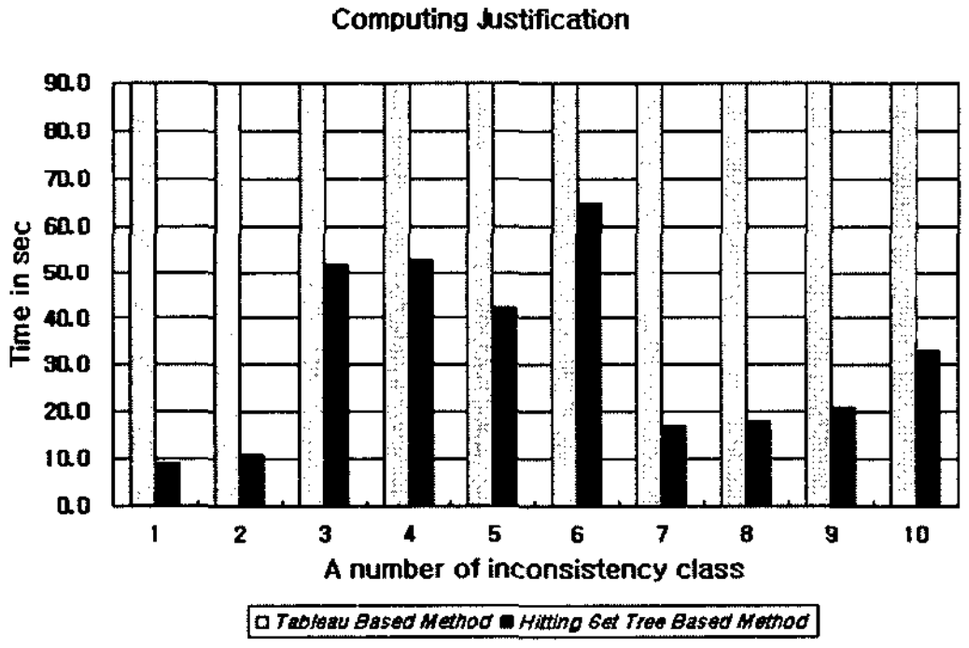

그림 8 Food 온톨로지에서 논리적 오류를 일으키는 클래스 수에 따른 MEXS 구성시간

로 적지 않은 시간이 소모되는 것으로 분석된다.

표 3은 MEXS 구성의 정확도의 실험 결과를 보여준다. 결과를 종합해 보면 논리적 오류를 유발하는 공리 집합을 제거할 경우 온톨로지 자체에 논리적 오류는 사라지지만, 실제 Subsumption 추론의 성능을 많이 떨어트린다. 이것은 MEXS를 구성하는 공리들은 논리적 오류의 근원 공리뿐 아니라 논리적 오류를 유발하는 원인 공리들을 모두 저장하기 때문에 실제 MEXS를 구성하는 공리들이 논리적 오류를 일으키지 못한다. 따라서 단순히 MEXS를 제거하는 것은 Subsumption 추론에는 많은 부정적인 영향을 미친다. 따라서 MEXS를 구성하는 것 이 외에도 논리적 오류의 근원 공리를 찾는 것이 매우 중요하며, 이 부분은 향후 계속 연구되어질 것이다.

8. 결 론

본 논문에서는 논리적으로 정당하지 못한 온톨로지를 디버깅 하기위한 MEXS(Minimum Expression Axiom Set) 추출과 저장에 대한 기법을 제안한다. MEXS를 추출하기 위해서는 온톨로지 내에서 논리적인 오류를 유발하는 공리들을 찾아내는 방법은 매우 중요하다고 할 수 있다. 따라서 Tableaux 알고리즘 기반의 MEXS 구성 기법과 이를 좀 더 최적화한 Hitting Set Tree 기반의 MEXS 구성 기법을 제안하였다. 이러한 기법들은 온톨로지 내에서 논리적 오류를 유발하는 부분들만을 정확하게 지적해주기 때문에 온톨로지 구축자가 대용량 온톨로지를 디버깅하기 위해 많은 도움을 줄 수 있다. 또한 클래스간의 Subsumption 추론 과정을 표현하는데 좋은 자료가 된다. 현재 제안된 방법을 실험하기 위해서 기존 온톨로지 추론 엔진인 Pellet을 사용하였기 때문에 실험결과는 많은 부분에서 Pellet의 성능에 의존된다. 따라서 향후에는, 좀 더 MEXS를 효과적으로 구성하는 추론엔진개발과 더불어 MEXS기반의 온톨로지 디버깅 Planning 기법에 대한 연구가 진행될 것이다.

참 고 문 헌

[1] Berners-Lee, T., Hendler, J. and Lassila, O., "The Semantic Web," Scientific American, 2001.
[2] Gruber, T., "A translation approach to portable ontologies," Knowledge Acquisition, Vol. 5, No. 2, pp. 199~220, 1993.
[3] 최중민, "시맨틱 웹의 개요와 연구 동향," 정보과학회지, Vol.21, No.03, pp. 4~10, 2003.03.
[4] Grigoris, A., Frank van Harmelen, A Semantic Web Primer. The MIT Press, 2004.
[5] Baader F., et al, The Description Logic Handbook, Cambridge, 2003.
[6] Genesereth, M., Nilsson, N. Logical Foundations of Artificial Intelligence, Morgan Kaufman, 1987.
[7] Ian Horrocks, Optimizing Tableaux Decision Procedures for Description Logics, PhD thesis, University of Manchester, 1997.
[8] Steffen Staab, Rudi Studer (Eds.): Handbook on ontologies: International Handbooks on Information Systems. Springer 2004.

김 제 민

2001년 숭실대학교 컴퓨터학과(학사). 2004년 숭실대학교 대학원 컴퓨터학과(석사) 2004년~현재 숭실대학교 대학원 컴퓨터학과 박사과정. 관심분야는 인공지능, 시맨틱 웹, 유비쿼터스 컴퓨팅

박 영 택

1978년 서울대학교 전자공학과(학사). 1980년 KAIST 전산학(석사). 1992년 Univ. of Illinois at Urbana Champaign(박사). 1981년~현재 숭실대학교 컴퓨터학과 교수. 관심분야는 인공지능, 에이전트, 전문가 시스템, 시멘틱 웹