태블로 알고리즘 기반 온톨로지 추론 엔진의 속도 향상을 위한 방법

(Methods to Reduce Execution Time of Ontology Reasoners based on Tableaux Algorithm)

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

요 약 온톨로지의 크기가 대형화됨에 따라, 온톨로지 내부 구조는 점점 복잡해지고 있다. 따라서 온톨로지 구축과정에서 발생하는 여러 가지 논리적 오류를 찾아내어 수정하는 것은 매우 어려운 작업이 되고 있다. Minerva[1]는 OWL로 작성한 온톨로지 중 논리적 오류를 갖는 개념들을 자동으로 탐지하고, 개념간의 계층 관계를 추론하기 위해 개발된 온톨로지 추론 엔진이다. Minerva를 포함한 대부분의 서술 논리 기반의 온톨로지 추론 엔진은 태블로 알고리즘(Tableau Algorithm)을 기반으로 동작한다. 태블로 알고리즘을 그대로 적용할 경우 시간 및 공간 복잡도가 상당히 높아지기 때문에 다양한 최적화 기법이 필요하다. 본 논문에서는 태블로 알고리즘을 사용하는 온톨로지 추론 엔진의 속도를 향상시키는 최적화 기법들을 제안한다. 제안한 기법들은 선행 연구로서 이미 개발된 온톨로지 추론엔진 Minerva에 적용되어 성능향상을 이끌어 내었다.

키워드 : 온톨로지, 태블로 알고리즘, 온톨로지 추론엔진, 최적화 기법

Abstract As size of ontology has been increased more and more, the descriptions in the ontologies become more complicated. Therefore finding and modifying unsatisfiable concepts is hard work in ontology construction process. Minerva is an ontology reasoner which detects unsatisfiable concepts automatically and infers subsumption relation between concepts in ontology. Most description logic based ontology reasoners (including Minerva) work using tableaux algorithm. Because tableaux algorithm is very costly, ontology reasoners need various optimization methods. In this paper, we propose optimizing methods to reduce execution time of tableaux algorithm based ontology reasoner. Proposed methods were applied to Minerva which was developed as preceding study result. In consequence the new version Minerva shows high performance.

Key words : Ontology, Tableaux Algorithm, Ontology Reasoner, Optimized Method

1. 서 론

차세대 웹 환경인 시맨틱 웹에서 중추적인 역할을 하는 것은 기계가 이해할 수 있도록 개념들을 명확하게 명시하고, 이들 개념들을 공유할 수 있는 형식으로 표현한 온톨로지다. 온톨로지의 크기가 대형화되면서, 온톨로지의 내부 구조는 점점 복잡해지고 있다. 따라서 온톨로지에 논리적 오류가 발생할 경우 온톨로지 개발자들이 이러한 오류들을 찾아내어 수정하는 것은 매우 어려운 작업이 되고 있다. 현재 온톨로지의 논리적 오류와 온톨로지를 구성하는 개념들 간의 계층 관계를 추론하기 위해서, OWL 추론 엔진들이 소개되고 있다[2-4]. 이중 현재 많이 활용되고 있는 OWL 기반의 온톨로지 추론 엔진은 Ian Horroks에 의해 만들어진 FaCT(Fast Classification of Terminology)[2]와 FaCT++이며, 최근에는 Pellet[3]과 KAON2[5]의 활용도 점차 증가되고 있는 추세다. 국내에서 만들어진 대표적인 온톨로지 추

· 본 논문은 숭실대학교의 지원을 받았습니다.

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

논문접수 : 2008년 10월 28일
심사완료 : 2008년 12월 18일

정보과학회논문지 : 소프트웨어 및 응용 제36권 제2호(2009.2)

론엔진으로는 보쌈[6]과 본 논문의 선행 연구(온톨로지 추론 엔진 구축)의 결과인 Minerva[1]가 있다. 두 엔진의 차이점은 보쌈이 오직 정확한(sound) 추론 결과만을 산출하는 반면, Minerva는 정확하면서 완전한(complete) 결과를 산출한다.

Minerva를 포함한 대부분의 서술 논리 기반의 온톨로지 추론 엔진은 태블로 알고리즘(Tableaux Algorithm)을 기반으로 동작한다. 그러나 태블로 알고리즘을 그대로 적용할 경우, 추론 결과를 산출하기 위한 시간 및 공간 복잡도가 상당히 높아지기 때문에 이를 줄이기 위한 다양한 최적화 기법이 필요하다.

본 논문에서는 태블로 알고리즘 기반의 온톨로지 추론을 위한 최적화 기법 3가지를 제안한다. 첫 번째는 온톨로지를 구성하는 개념들에 대한 논리적 정당성 검사 횟수를 최소화 하기위한 확장된 캐시 기법-논리적 개념 캐시(satisfiable concept cache), 두 번째는 태블로 알고리즘에 의해 생성되는 노드의 블로킹 여부를 보다 신속하게 검색하는 향상된 블로킹 노드 검색 기법, 마지막은 태블로 알고리즘을 구성하는 규칙 중 분기 선택(non-deterministic) 문제를 발생시키는 ⊔-규칙 호출을 최소화하는 논리합 배열 기법(disjunction order) 이다. 제안한 기법들은 선행 연구로서 이미 개발된 온톨로지 추론엔진인 Minerva에 적용되어 일부 온톨로지에 대해서 상당한 성능 향상을 이끌어 내었다.

2. 기본 개념

2.1 태블로 알고리즘(Tableaux Algorithm)

대부분의 서술 논리 기반 온톨로지 추론엔진들은 태블로 알고리즘을 기반으로 구축되었다. 태블로 알고리즘은 초기 조건에 해당되는 규칙을 실행시켜 레이블과 노드를 확장해 나가는 방식으로서, 확장되어지고 있는 레이블에 논리적 충돌(clash)이 발생하면, 논리적으로 정당하지 못한 것이라고 판정한다. 반면 더 이상 규칙을 적용할 수 없을 때까지 논리적 충돌이 발생하지 않으면 논리적으로 정당하다고 판정한다. 현재 서술 논리 기반의 많은 온톨로지 추론 기술은 태블로 알고리즘을 기반으로 하고 있다.

태블로 알고리즘의 기본 아이디어는 증명하고자 하는 내용의 논리적 부정(negation)에 대해서 다양한 변환 규칙을 적용하여 정당하지 않다는 것을 보여주는 방식을 취하고 있다. 서술 논리는 ⊓, ⊔, ¬, ∃, ∀등으로 표현되기 때문에 태블로 알고리즘에서는 이에 대한 확장 규칙을 반복적으로 적용하면서 탐색공간을 확장하게 된다. 그리고 모든 탐색 공간의 말단 노드가 모순(contradiction)이라는 것을 보여줌으로써 증명하고자 하는 내용이 정당하다는 것을 증명하게 된다. 노드가 모순이 되기 위해서는 같은 노드 레이블에 개념 C와 부정인 ¬C가 동시에 존재해야 한다. 다음 표 1은 태블로 알고리즘에서 활용하는 기본 확장 규칙을 보여준다.

표 1 태블로 알고리즘

⊓-rule	if 1. $(C_1 \sqcap C_2) \in \mathcal{L}(x)$ 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. $(C_1 \sqcup C_2) \in \mathcal{L}(x)$ 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 $\mathcal{L}(y) \rightarrow \mathcal{L}(y) \cup \{C\}$

표 1에 기술된 각 규칙의 의미는 다음과 같다.

- ⊓-규칙 : $C_1 \sqcap C_2$가 논리적으로 정당하기 위해서는 C_1과 C_2가 논리적으로 정당해야 되기 때문에 C_1과 C_2를 같은 노드 레이블에 추가
- ⊔-규칙 : $C_1 \sqcup C_2$가 논리적으로 정당하기 위해서는 C_1과 C_2중의 하나는 반드시 논리적으로 정당해야 되기 때문에 먼저 C_1을 레이블에 추가. 만약 논리적 충돌이 발생하면 레이블을 원상태로 환원한 후 C_2를 레이블에 추가
- ∃-규칙 : $\exists R.C$가 논리적으로 정당하기 위해서는 레이블 안에 $\exists R.C$를 갖는 노드가 C의 개체가 되는 노드를 대상으로 최소한 1개 이상의 R관계를 가져야하기 때문에 R관계가 하나도 없다면 임의의 노드를 생성한 후, R관계로 연결하고, 노드 레이블에 C를 추가
- ∀-규칙 : $\forall R.C$가 논리적으로 정당하기 위해서는 레이블 안에 $\forall R.C$를 갖는 노드가 R관계로 항상 C의 개체와 연결되어야 하기 때문에, R관계가 존재한다면 R관계로 연결된 모든 노드 레이블에 C를 추가

3. 선행 연구

[2,3,7]에는 태블로 알고리즘 기반의 온톨로지 추론을 위한 여러 가지 최적화 기법이 소개되어 있다. 본 논문에서 제안하는 기법은 이미 발표된 것 외에 추가적인 것으로서 태블로 알고리즘의 속도를 향상시킨다는 공통점이 있다. 본 장에서는 본 연구의 선행 연구를 소개하고, 이미 [2,3,7]에서 발표한 최적화 기법 외에 추가로 적용한 최적화 기법을 설명한다.

3.1 온톨로지 추론 엔진-Minerva

Minerva는 서술 논리(description logic) 기반의 OWL 온톨로지 추론엔진이며, 현재 SHIQ 수준의 표현력을 지닌 온톨로지의 스키마(T-Box) 추론 기능을 지원한다. Minerva는 온톨로지에 논리적 오류가 존재하는지를 검사하고, 포함관계 추론을 통해 온톨로지를 구성하는 개념들 간의 계층 구조를 계산한다.

3.1.1 온톨로지의 논리적 오류 검사 과정

온톨로지의 논리적 오류 검사는 온톨로지를 구성하는 개념들에 대해서 각각 논리적 검증을 실행하는 것이다. 각 개념의 논리적 검증은 태블로 알고리즘을 진행시키면서, 해당 규칙을 하나씩 적용한다. 이때 규칙을 적용하는 순서가 매우 중요하다. Minerva는 스케줄링 큐인 To-do 리스트를 적용하여 적용되는 규칙의 순서를 지정한다. To-do 리스트는 규칙이 적용될 개념과 그 개념이 위치한 노드를 기록하고 있는 큐들의 집합으로서, 각각의 큐는 서로 우선순위를 가지고 있다. 따라서 태블로 알고리즘이 진행되면서 노드가 생성되는 경우, 규칙을 적용할 개념들이 노드 레이블에 존재하는지를 확인하기 위해 모든 노드를 검사할 필요가 없다. Minerva는 검증하고자하는 각 개념에 대해 태블로 알고리즘을 진행하면서, 논리적 모순(contradiction)이 발생(같은 노드 레이블에 개념C와 부정인 ¬C가 동시에 존재)하면 논리적 오류를 갖는 개념을 캐싱하고, 이를 분류하여 인터페이스를 통해 온톨로지 구축자에게 보이게 된다. 이때 검증을 마친 개념들에 대해서 차후에 포함관계 추론 과정을 보다 빨리 수행하기 위한 의사 모델(pseudo model)[9]을 구축하게 되는데, 의사 모델은 태블로 알고리즘이 진행되면서 생성된 트리의 루트 노드 레이블 정보로 구성된다.

3.1.2 포함관계 추론 과정

Minerva에서의 포함 관계 추론은 두 개념간의 암시적 계층관계를 추론하는 것이다. 일반적으로 포함관계 추론을 시도하는 횟수는 n개 개념이 존재할 때 (n^2-n)번이다. 포함관계 검증을 위한 태블로 알고리즘은 많은 시간을 소모하므로, 온톨로지에 명시된 계층 구조를 이용하여 미리 순서리스트를 만들어 두면 포함관계 검증 횟수를 줄일 수 있다. Minerva에서 순서리스트 역할을 해주는 것이 정의 순서 리스트(definition order)인데, 정의 순서 리스트는 온톨로지 상의 두 개념 x와 y가 x ⊑y의 관계를 가질 때, x<y의 순서로 개념들을 나열해 준다. 정의 순서 리스트를 이용한 포함관계 검증 횟수는 n(n+1)/2번이다. 정의 순서리스트가 정해지면 리스트에 저장된 개념들을 바탕으로 정렬된 순서에 따라 태블로 알고리즘(2.1절에서 설명)을 이용하여 포함 관계를 추론하고 이를 바탕으로 새로운 계층 구조를 구축한다.

Minerva에서 두 개념의 포함관계 C⊑D를 증명하기 위해서는, 이것의 논리 부정인 C⊓¬D가 정당하지 못하다는 것을 보이면 된다. 일반적으로 태블로 알고리즘은 많은 시간과 공간을 소모한다. 따라서 C⊓¬D에 대해서 태블로 알고리즘을 적용하여 정당성을 테스트하기에 앞서, 논리적 오류 검사 과정에서 생성한 C와 ¬D개념의 의사 모델[8]을 하나의 노드로 합병하여, 논리적 모순이 발생하는지 확인한다. 만약 논리적 모순이 발생한다면, 태블로 알고리즘 적용 없이 논리적으로 정당하지 않다고 판단할 수 있다.

3.2 Minerva에 적용된 최적화 기법

3.2.1 비논리적 개념 캐시

Minerva는 온톨로지의 논리적 오류를 찾아내기 위해서 태블로 알고리즘을 통해 모든 개념들을 하나씩 검사한다. 비논리적 개념 캐시는 특정 개념에 논리적 오류가 발생하였을 경우 논리 확장 맵(unfold-map)에 존재하는 그 개념의 논리 확장 정보를 거짓(bottom)으로 바꿈으로서, 또 다른 개념을 검사할 때, 이미 검사되어 거짓으로 기록된 개념을 논리 확장해야 하는 상황이 발생하면, 더 이상의 알고리즘의 진행을 중단하고 비논리적 개념임을 판정하게 된다. 예를 들어, 논리 확장 맵에 A::Q⊓C, B::A⊓...⊓Z라는 정보가 존재할 경우, 개념 A에 논리적 오류가 발견되면 개념 A의 논리 확장 정보를 A::bottom으로 재 기록한다. 다음 개념 B의 논리적 오류를 검사할 때, 태블로 알고리즘을 통해 개념 B를 논리 확장하면 개념 A가 추가되며, 또 다시 개념 A를 논리 확장해야 하는데, 이미 개념 A는 거짓으로 판명되었다는 기록이 논리 확장 맵에 존재하기 때문에, 결국 개념 B 역시 논리적 오류를 갖는 개념이 된다.

3.2.2 개선된 상-하위 검색과 하위-상 검색

태블로 알고리즘을 적용하여 온톨로지의 새로운 계층 구조를 구축하며 개념들 간의 포함관계를 검색하는 대표적인 알고리즘은 상-하위(top-down) 검색 알고리즘과 하위-상(bottom-up) 검색 알고리즘[7]이다. 상-하위 검색 알고리즘은 특정 개념의 상위 개념들을 검색해주는 알고리즘이고 하위-상 검색 알고리즘은 특정 개념의 하위 개념들을 찾아주는 알고리즘이다. 이 두 알고리즘은 두 개념의 포함 관계 검증을 기반으로 작업이 진행된다.

상-하위 검색은 "하나의 개념이 또 다른 개념에 포함되기 위해서는 그것의 상위 개념에 이미 포함되어야 한다." 라는 수학적 집합 원리에 근거를 두고 있다. 따라서 상-하위 검색은 포함 관계 검증을 통해 특정 개념이 또 다른 개념에 포함되지 않으면, 그것의 하위 개념들과 더 이상 포함 관계 검증을 실시하지 않는다.

하위-상 검색은 "하나의 개념이 또 다른 개념을 포함

하기 위해서는 그것의 하위 개념들을 모두 포함해야 된다."라는 수학적 집합 원리에 근거를 두고 있다. 따라서 하위-상 검색은 포함 관계 검증을 통해 특정 개념이 또 다른 개념을 포함하지 않으면 그것의 상위 개념과 더 이상 포함 관계 검증을 실시하지 않는다.

포함 관계 검증은 많은 연산 시간을 소모하는 태블로 알고리즘을 사용하기 때문에 그 횟수를 줄이는 것은 추론 엔진 성능에 막대한 영향을 끼친다고 할 수 있다. 따라서 검색 알고리즘을 더욱 최적화 시켜서 포함 관계 검증의 횟수를 최소화 시키는 것은 매우 중요하다. 이를 위해 Minerva의 검색 알고리즘[9]에는 다음 4가지 개념이 적용되었다.

- 부정정보의 하위 전파(negative information down propagation) - 상-하위 검색에서 특정 개념이 또 다른 개념에 포함되지 않으면, 그것의 하위 개념들과 더 이상 포함 관계 검증을 실시할 필요가 없다. 따라서 포함 관계 검증이 필요 없는 하위 개념들 모두에 부정정보(negative information)를 기록한다.

$$A \not\sqsubseteq B \text{ and } C \sqsubseteq B \text{ implies } A \not\sqsubseteq C$$

- 긍정정보의 상위 전파(positive information up propagation) - 상-하위 검색에서 특정 개념이 또 다른 개념에 포함된다면 그것의 상위 개념들에도 모두 포함되기 때문에, 차후에 그 상위 개념들과 다시 포함 관계 검증을 실시할 필요가 없다. 따라서 포함 관계 검증이 필요 없는 상위 개념들 모두에 긍정정보(positive information)를 기록한다.

$$A \sqsubseteq B \text{ and } B \sqsubseteq C \text{ implies } A \sqsubseteq C$$

- 부정정보의 상위 전파(negative information up propagation) - 하위-상 검색에서 특정 개념이 또 다른 개념을 포함하지 않는다면 그것의 상위 개념들도 포함 하지 않기 때문에 포함 관계 검증을 실시할 필요가 없다. 따라서 포함 관계 검증이 필요 없는 하위 개념들 모두에 부정정보(negative information)를 기록한다.

$$B \not\sqsubseteq A \text{ and } B \sqsubseteq C \text{ implies } C \not\sqsubseteq A$$

- 긍정정보의 하위 전파(positive information down propagation) - 하위-상 검색에서 특정 개념이 또 다른 개념을 포함한다면 그것의 하위 개념들 모두를 포함하기 때문에, 차후에 그 하위 개념들과 다시 포함 관계 검증을 실시할 필요가 없다. 따라서 포함 관계 검증이 필요 없는 하위 개념들 모두에 긍정정보(positive information)를 기록한다.

$$B \sqsubseteq A \text{ and } C \sqsubseteq B \text{ implies } C \sqsubseteq A$$

4. 태블로 알고리즘 기반 온톨로지 추론 엔진의 속도 향상을 위한 최적화 기법

4.1 논리적 개념 캐시

태블로 알고리즘은 온톨로지를 구축하는 각각의 개념에 대해 반복적인 규칙 적용을 통해 모델을 구축하면서 논리적 정당성을 판단하기 때문에 시간을 많이 소모하게 된다. 온톨로지의 논리적 오류 검사를 위해 태블로 알고리즘이 적용되는 개념의 개수를 최소화하는 것은 매우 중요하다. 따라서 다음과 같은 수학적 개념을 적용하여 태블로 알고리즘이 적용되는 개념의 개수를 최소화하는 논리적 개념 캐시를 Minerva에 적용하였다.

$$(C_1 \sqsubseteq C_2) \subset TBox$$

If C_2 is satisfiable then C_1 is satisfiable

논리식 A $\sqsubseteq$ (B$\sqcap$C)가 주어졌을 때 태블로 알고리즘을 사용하여 개념 A의 논리적 정당성을 검사하기 위해서는 A의 상위 개념 (B$\sqcap$C)를 레이블에 추가하여 (B$\sqcap$C)의 논리적 정당성을 검사한다. $\sqcap$-규칙에 의해 B와 C가 레이블에 추가된 후, 논리적 오류가 발생하지 않으면서 더 이상 적용할 태블로 규칙이 없다면 최종적으로 개념 A는 논리적으로 정당하게 된다. 즉, B$\sqcap$C가 논리적으로 정당하다면 A역시 논리적으로 정당하다. 다른 관점에서 보면 결국 개념 A가 논리적으로 정당하다는 것은 A의 상위 개념 B와 C 역시 논리적으로 정당하다는 것을 의미하므로, 차후에 B와 C에 대해서 논리적 정당성 검사를 할 필요가 없다.

본 논문에서 제안하는 논리적 개념 캐시의 역할은 특정 개념의 논리적 정당성을 검사하기 위해 레이블에 추가 되는 모든 개념들을 해시에 키로 등록하고, 만약 검사 중인 특정 개념이 논리적으로 정당하면, 해시에 등록된 키의 값을 전부 논리적 정당성을 의미하는 불리언 값으로 기록함으로써, 태블로 알고리즘이 적용되는 개념의 개수를 최소화한다.

4.2 향상된 블로킹 노드 검색 기법

태블로 알고리즘을 구성하는 규칙 중 존재 한정 규칙($\exists$-규칙)은 모델에 새로운 노드를 생성하는 규칙이다. 따라서 무한적으로 노드를 반복 생성할 가능성을 가진다. 예를 들어 TBox안에 A $\sqsubseteq$ $\exists$R.B와 B $\sqsubseteq$ $\exists$R.A가 존재할 경우, 개념 A의 논리적 정당성을 증명하기 위해서 간선 R로 연결되는 노드가 반복적으로 생성된다(2.1절의 표 1 참조). 이러한 현상을 방지하기 위해 무한 반복될 가능성이 있는 노드를 블로킹한다. 노드의 블로킹 조건은 X′, Y, Y′이 노드 X의 조상 노드이고, X는 X′의 계승노드(successor), Y가 Y′의 계승 노드일 때, X의 레이블의 내용과 Y의 레이블의 내용이 동등한 경우이다. 이러한 경우, 노드 X는 블로킹 되어 더 이상 존재 한정 규칙을 적용받지 않는다.

일반적으로 노드의 블로킹 여부를 검사할 경우 현재 노드의 모든 조상 노드와 레이블 비교를 한다. 따라서

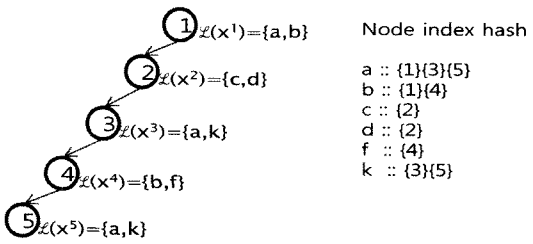


그림 1 특정 노드의 블로킹 여부를 검사하기 위한 노드 색인 해시

표 2 블로킹 노드 검색 의사 코드

```
procedure Node Index Hash Generation
   input: current concept<list>
            node index hash<map>
            current node index<set element>
  element a ← current node index
  if current concept exist in node index hash
      add a in node index hash<current concept>
  else
      generate key using current concept
      add a in node index hash<current concept>

procedure Directed Blocking Detection
   input: concepts in label<list>
            node index hash<map>
            current node index<set element>
  set A ← label[0]
  for(int i=1; i<size of label; i++)
       set B ← label[i]
       set A ← set A ∩ set B
  set A ← set A - current node index
  if   set A is not empty && compare label(set A, label)
       return true
  else
       return false
```

태블로 모델의 깊이(depth)가 커질수록 노드의 상태를 검사하는 데 시간을 많이 소모하게 된다.

본 논문에서는 노드의 상태를 검사하는 데 소요되는 시간을 줄이기 위해서 노드 색인 해시를 이용한 블로킹 검사 기법을 설계하여 Minerva에 적용하였다. 노드 색인 해시는 레이블에 추가되는 개념을 키로 저장하고, 개념이 존재하는 레이블을 갖는 노드 번호들을 집합으로 묶어 값으로 저장한다. 예를 들어, 그림 1은 노드 색인 해시의 구조를 보여주는데, 태블로 모델 안에서 개념 a는 1,3,5번 노드의 레이블에 존재한다. 따라서 맨 처음 1번 노드에 a가 추가될 때 a를 해시에 키로 등록하고 공집합(empty set)을 값으로 설정한 후 노드 번호 1을 집합에 추가한다.

제안하는 블로킹 검사 기법은 노드 색인 해시를 사용함으로서, 현재 노드 레이블과 모든 상위 노드 레이블을 비교할 필요 없이, 현재 레이블에 존재하는 개념과 대응하는 해시 값(집합)들의 교집합을 구성해 봄으로써, 반드시 비교해야 하는 노드 레이블을 바로 찾아 낼 수 있다. 그림 1을 기준으로 예를 들어보면, 5번 노드의 상태를 검사하기 위해서, 5번 노드의 레이블에 존재하는 개념 a와 k의 해시 값들의 교집합을 구성해본다. 그 결과로 집합 {3,5}가 구성되며, 현재 노드 번호인 5를 제거할 경우 {3}이 남게 된다. 따라서 현재 노드와 3번 노드의 레이블 내용만 비교하면 블로킹 여부를 판단할 수 있게 된다. 표 2는 노드 색인 해시를 이용하여 노드의 블로킹 여부를 판단하는 의사코드다.

4.3 논리합 배열 기법(Disjunction Order)

태블로 알고리즘을 구성하는 규칙들 중 논리합 규칙(⊔-rule)은 논리합을 구성하는 첫 번째 개념을 레이블에 추가한 후, 논리적 모순이 발생하면 그 개념이 추가되기 전 상태로 레이블을 환원시킨 후, 논리합을 구성하는 두 번째 개념을 레이블에 추가한다. 이러한 과정은 레이블에 논리합을 구성하는 특정 개념을 추가했을 때 논리적 모순이 발생하지 않거나(논리적 정당), 논리합을 구성하는 모든 개념들에 대해 논리적 모순이 발생(논리적 모순)해야 중지 된다. 이 경우, 논리합을 구성하는 개념들이 논리적 모순을 유발하지 않더라도 다른 규칙에 의해 레이블에 추가된 개념이 논리적 모순을 유발한다면 불필요한 분기(branch)가 생성되는 분기 선택 문제가 발생된다. 따라서 논리합 규칙의 호출을 최소화하는 것은 추론엔진의 성능 향상의 중요한 요인이다.

본 논문에서는 논리합-규칙 호출을 최소화하기 위해 논리합 배열 기법(disjunction order)을 설계하여 Minerva에 적용하였다. 논리합 배열 기법에 사용된 개념은 흡수 법칙(absorption law)과 논리합 배열 캐시이며, 자세한 내용은 다음과 같다.

4.3.1 흡수 법칙

A ⊓ (A ⊔ D) ≡A일 때

- A가 사실이면 (A ⊔ D)의 결과에 상관없이 전체는 사실
- A가 거짓이면 (A ⊔ D)의 결과에 상관없이 전체는 거짓

흡수 법칙을 적용한 논리합 배열 기법은 논리합으로 연결된 개념들을 배열로 정렬한 후, 배열을 구성하는 개념들 중 레이블에 존재하는 것이 하나라도 있다면 분기(branch)를 생성하지 않는다. 즉, 논리합 규칙이 확장되지 않는다. 예를 들어, 아래 그림 2를 보면, 개념 A의 논리적 정당성을 테스트하기 위해, A의 논리 확장 개념인 {B, (C⊔D⊔B)}가 레이블에 추가된다. 논리 확장이란 특정 개념의 논리적 정당성을 검사하기 위해 상위 개념들의 논리적 정당성을 검증하는 것이다. 즉 상위 개념이 논리적으로 정당해야 그 하위 개념 역시 논리적으로 정당할 수 있다. 따라서 그림 2의 논리 확장 규칙(unfolding-rule)은 특정 개념이 레이블에 추가될 경우

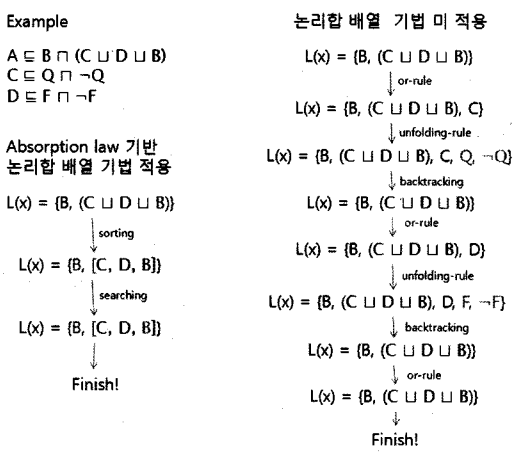


그림 2 흡수 법칙 기반 논리합 배열 기법 적용 예

그 개념의 상위 개념들도 레이블에 추가한다.

흡수 법칙을 적용한 논리합 배열을 적용하게 되면, 논리합 (C⊔D⊔B)를 구성하는 개념들 중 개념 B는 이미 레이블에 존재하므로 논리합 규칙과 추가적인 태블로 규칙 적용 없이 A가 논리적으로 정당하다는 결과를 보여준다. 반면에 논리합 배열을 적용하지 않을 경우, C와 D가 논리적 충돌(contradiction)을 발생시키므로, 3번의 논리합 규칙을 호출하지만 결과적으로는 A가 논리적으로 정당하다는 결과를 보여준다.

4.3.2 논리합 배열 캐시

$A \sqsubseteq (B_1 \sqcup C_1 \sqcup D_1 \sqcup E_1 \sqcup F_1 \sqcup G_1) \sqcap \ldots \sqcap (B_4 \sqcup C_4 \sqcup D_4 \sqcup E_4 \sqcup F_4 \sqcup G_4)$

$B \sqsubseteq A$

$C \sqsubseteq B$

$S_1 \sqsubseteq C$

...

$S_n \sqsubseteq C$

위와 같은 공리가 T-Box에 존재할 경우, 개념 A가 논리적 충돌을 발생시키지 않으면, A의 하위 개념인 B, C, S_1, ..., S_n의 논리적 정당성을 검사하기 위해서 불필요한 논리합 규칙을 여러 번 호출하게 된다. 물론 상위 개념에서 논리합으로 연결된 개념이 논리적 충돌을 일으키지 않는다고 해서, 하위 개념이 논리적 충돌을 일으키는데 전혀 영향을 주지 않는 것은 아니다. 예를 들어, 개념 B의 논리 확장 정보가 $\neg(B_1 \sqcup C_1 \sqcup D_1 \sqcup E_1 \sqcup F_1 \sqcup G_1) \sqcap \ldots \sqcap \neg(B_4 \sqcup C_4 \sqcup D_4 \sqcup E_4 \sqcup F_4 \sqcup G_4)$라고 가정 한다면 상위 개념에서의 논리합으로 연결된 개념과 B의 논리 확장 정보는 논리적 충돌을 일으키게 된다. 이와 같은 사실과 4.3.1에서 설명한 흡수 법칙을 고려하여 논리합 배열 캐시 알고리즘을 설계하여 Minerva에 적용하였다.

먼저 특정 개념의 논리합 규칙을 적용한 후 논리적 정당성을 일으키는 조합을 캐싱한다. 이 때 논리합으로 연결된 개념을 키로 설정하고 논리적 정당성을 일으키는 조합을 값으로 기록한다. 예를 들어, 개념 A의 $(B_1 \sqcup C_1 \sqcup D_1 \sqcup E_1 \sqcup F_1 \sqcup G_1) \sqcap \ldots \sqcap (B_4 \sqcup C_4 \sqcup D_4 \sqcup E_4 \sqcup F_4 \sqcup G_4)$에 논리합 규칙을 반복적으로 적용하여 논리적 정당성을 이끌어내는 조합이 $\{B_1, C_2, D_3, E_4\}$라고 한다면, 논리합 배열 캐시의 키는 $(B_1 \sqcup C_1 \sqcup D_1 \sqcup E_1 \sqcup F_1 \sqcup G_1) \sqcap \ldots \sqcap (B_4 \sqcup C_4 \sqcup D_4 \sqcup E_4 \sqcup F_4 \sqcup G_4)$이고, 값은 $\{B_1, C_2, D_3, E_4\}$가 된다.

다음 하위 개념의 논리적 정당성을 검사할 때, 상위 개념으로부터 상속 받은 논리합 연결 개념에 대해서 논리합 규칙을 적용하기 전에 해당 논리합 연결 개념에 대한 논리합 배열 캐시 정보를 레이블에 추가한 후, 적용 되어야할 다른 규칙들을 확장해본다. 이때 논리적 충돌이 발생하지 않으면, 불필요한 논리합 규칙 없이 하위 개념은 논리적으로 정당하다는 결론을 내릴 수 있다. 만약 논리적 충돌이 발생할 경우 두 가지를 고려한다. 먼저 상위 개념으로부터 상속받은 논리합 연결 개념이 논리적 충돌에 영향을 주지 않을 경우, 추가적인 논리합 규칙 적용 없이 논리적으로 정당하지 않다는 결론을 내릴 수 있다. 반면 상속받은 논리합 연결 개념이 논리적 충돌에 영향을 줄 경우에는 논리합 규칙을 적용하면서 논리적 정당성 테스트를 수행한다.

5. 실험 및 평가

본 논문의 선행 연구로서 구현된 Minerva는 SHIQ 수준의 표현력을 지닌 온톨로지 스키마(T-Box) 추론 기능을 지원한다. 따라서 제안한 최적화 기법에 대한 성능 평가를 위해 Minerva가 제공하는 온톨로지의 논리적 오류 검사와 포함관계 추론에 대해 초점을 맞춰서 Minerva와 제안한 기법을 적용한 Minerva의 속도를 비교하였다.

본 실험은 펜티엄 CPU(1.77GHZ)와 메모리(1G)를 장착한 랩탑에서 실행하였으며, 실험에 적용할 데이타로서 mindswap에서 자신들의 프로젝트를 위해 배포한 온톨로지와 Minerva를 구축할 당시 사용했었던 베타 테스트용 온톨로지들을 사용하였다. 표 4는 본 실험에 사용한 온톨로지의 세부내용을 보여준다.

그림 3과 그림 4는 본 논문에서 제안한 최적화 기법을 적용했을 때 10개의 테스트 온톨로지에 대한 논리적 오류 검사와 포함관계 추론에 소요된 시간을 보여준다. 논리적 오류 검사의 경우 대부분의 온톨로지는 최적화 기법이 적용이 됐을 때도 속도차가 크게 나지 않았으나, Dice 온톨로지의 경우 5배, Design_patterns 온톨로지의 경우 2.5배 정도 속도가 향상되었다.

표 4 실험에 사용된 온톨로지 정보

출처	온톨로지	크기	클래스 개수	restriction 수
mindswap	dice	2.5M	528	55373
	buggy-sweet	483K	1537	1876
Minerva Beta Test Ontology	m_test01	9K	37	19
	m_test02	15K	108	44
	image	168K	212	658
	generalOntology	54K	143	161
	SUMO	555K	629	696
	ISO_util	130K	144	1086
	fly_base	389K	1327	1991
	Design_patterns	180K	148	832

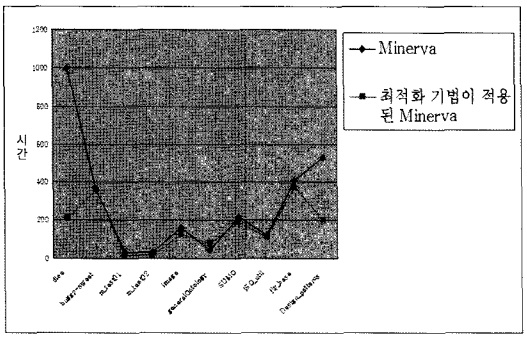


그림 3 테스트 온톨로지의 논리적 오류 검사에 소요된 시간

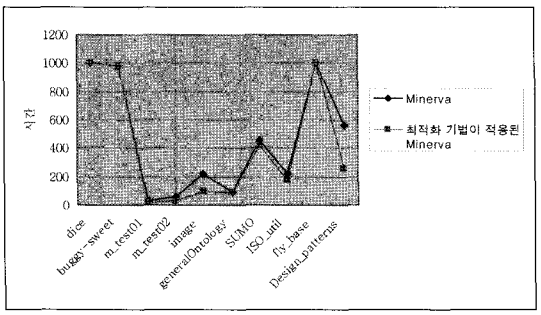


그림 4 테스트 온톨로지의 포함 관계 추론에 소요된 시간

Dice 온톨로지는 타 온톨로지에 비해 논리적 오류와 관련되지 않은 논리합이 많이 존재할 뿐 아니라 이렇게 많은 논리합으로 정의된 개념이 많은 개념들의 상위 개념이다. 따라서 Dice 온톨로지는 확장된 캐시 기법과 논리합 배열 기법을 통해 추론 속도가 향상 되었다. Design_patterns의 경우 역시 Dice 온톨로지와 같은 경우라고 할 수 있다. 특히 이 온톨로지는 존재 한정을 많이 사용하였다. 포함관계 추론의 경우 역시 대부분의 온톨로지는 최적화 기법이 적용이 됐을 때도 의사 모델의 적용 때문에 속도차가 크게 나지 않았으나, Image 온톨로지의 경우 2배, Design_patterns 온톨로지의 경우 2.8배 정도 속도가 향상되었다. Image 온톨로지의 경우 포함 관계 추론에서만 속도가 향상되었는데 이는 논리합 배열의 특성 때문인 걸로 분석 된다. 즉 두 개념의 포함 관계 C⊑D를 증명하기 위해, 이것의 논리 부정인 C⊓¬D가 정당하지 못하다는 것을 보이는 과정에서 C의 논리 확장 개념이 ¬A⊔¬D인 경우가 매우 많이 존재하였다. 따라서 본 논문에서 제안하는 최적화 기법은 모든 온톨로지에 대해서 속도 향상을 이끌어 내지는 않지만, 일부 온톨로지의 경우 2.5~5배 정도의 속도 향상을 이끌어 낼 수 있다.

6. 결론 및 향후 연구

태블로 알고리즘은 정확하고 완전한 추론 결과를 보여주지만, 추론 결과를 산출하기 위한 시간 및 공간 복잡도가 상당히 높기 때문에 이를 줄이기 위한 다양한 최적화 기법이 필요하다. 태블로 알고리즘의 이와 같은 단점을 해결하고자 본 논문에서는 태블로 알고리즘을 사용한 온톨로지 추론 엔진의 속도 향상을 위한 최적화 기법 3가지를 제안하였다. 첫 번째 기법은 온톨로지를 구성하는 개념들에 대한 논리적 정당성 검사 횟수를 최소화하기 위한 확장된 캐시 기법-논리적 개념 캐시(satisfiable concept cache), 두 번째는 태블로 알고리즘에 의해 생성되는 노드의 상태를 보다 신속하게 검색하는 향상된 블로킹 노드 검색 기법, 마지막은 태블로 알고리즘을 구성하는 규칙 중 분기 선택 문제를 발생시키는 논리합 규칙 호출을 최소화하는 논리합 배열 기법(disjunction order)이다. 선행 연구로서 이미 개발된 온톨로지 추론엔진인 Minerva에 본 논문에서 제안한 최적화 기법들을 적용한 결과 일부 온톨로지에 대해서 상당한 성능 향상을 이끌어 내었다.

본 논문에서 제안한 최적화 기법은 현재 *SHIQ* 수준으로 표현된 온톨로지 추론에 적합하다. 향후에 Minerva는 *SHOIQ* 수준까지 포함된 온톨로지를 추론 하도록 업그레이드될 것이며, 업그레이드된 Minerva의 추론 속도를 향상시키기 위한 최적화 기법들이 추가 연구될 예정이다.

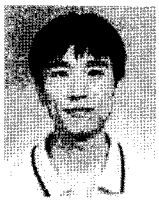
참 고 문 헌

[1] 김제민, 권순현, 박영택, "태블로 알고리즘 기반 온톨로지 추론 엔진", 정보과학회 KCC 2008.

[2] Evren Sirin, Bijan Parsia, Bernardo Cuenca Grau, Aditya Kalyanpur and Yarden Katz. Pellet: A practical OWL-DL reasoner, Journal of Web Semantics, 5(2), 2007.

[3] Dmitry Tsarkov and Ian Horrocks, FaCT++ description logic reasoner: System description, In Proc.

of the Int. Joint Conf. on Automated Reasoning, IJCAR, 2006.
[4] V. Haarslev and R. Möller. Racer: A Core Inference Engine for the Semantic Web. In Proceedings of the 2nd International Workshop on Evaluation of Ontology-based Tools(EON2003), located at the 2nd International Semantic Web Conference ISWC 2003, Sanibel Island, Florida, USA, October 20, pages 27-36, 2003.
[5] U. Hustadt, B. Motik U. Sattler. Reasoning in Description Logics with a Concrete Domain in the Framework of Resolution. Proc. of the 16th European Conference on Artificial Intelligence (ECAI 2004), August, 2004, Valencia, Spain, pp. 353-357.
[6] Min-su Jang, Joo-chan Sohn, Bossam: an extended rule engine for OWL Inferencing, Proceddings of RuleML 2004(LNCS Vol.3323), Nov. 8, 2004.
[7] Franz Baader, Bernhard Hollunder, Bernhard Nebel, Hans-Jurgen Profitlich, An Empirical Analysis of Optimization Techniques for Terminological Representation System, Principles of Knowledge Representation and reasoning - Proceedings of the 3th International Conference, October 1992, Cambridge, MA.
[8] F. Baader andW. Nutt. Basic description logics. In Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors, *The Description Logic Handbook: Theory, Implementation, and Applications*, pages 43-. 95. Cambridge University Press, 2003.
[9] 권순현, 김제민, 박영택, "온톨로지 포함관계 추론을 위한 최적화된 검색방법", 정보과학회 KCC 2008.



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