온톨로지 에디팅, 통합, 추론을 위한 관리도구: 초보적 디자인 그리고 구현

박경모°, 김동진
경희대학교 동서의료공학과 한의지식 연구실
saenim@khu.ac.kr°, sulmoiho@naver.com

Ontology Management Tool for Editing, Integration and Inference: Primitive Design and Implementation

Kyungmo Park°, Dongjin Kim
Department of Biomedical Engineering, Kyunghee university

요 약

온톨로지를 효율적으로 관리하기 위한 관리도구의 프레임워크로써, 온톨로지 편집과 통합, 추론을 다루고 있다. 온톨로지 편집에 있어 텍스트 기반의 편집과 한국어 사용자를 위한 편집 방식 지원 그리고 semi-Automatic 편집 및 템플릿 생성에 대해 다루고 있으며, 온톨로지 통합과 관련된 여러 알고리즘을 구상하고, 온톨로지에 대한 정당성 체크 및 향후 유비추론을 위한 프레임워크 개발에 대한 통합 측면과 추론에 대해 언급한다.

1. 서 론

지식표현 도구로써의 온톨로지 사용이 증가함에 따라 온톨로지를 효과적으로 관리할 수 있는 도구 개발에 대한 요구 또한 늘어나고 있다. 대표적인 온톨로지 구축 및 관리도구로 Protege[1]와 Oiled[2]가 있으며, Protege는 기본 모듈로 온톨로지 구축 및 에디팅, 추론을 지원하고 있다. 그러나 기존의 많은 관리도구들은 주로 단일 온톨로지만을 다루고 있어 다중 온톨로지 구축 및 편집이 어려우며, 편집과 관련해 버튼과 다이얼로그만을 바탕으로 하는 작업은 번거로운 점이 많다. 또한 DL(Description Logic)에 바탕한 온톨로지 개념의 구축 및 편집에 대한 표현식은 대부분의 일반 사용자에게는 익숙하지가 않다. 통합적 측면에 있어 어플리케이션에 속박된 단일화된 통합 방식은 사용자의 요구를 충족시키기에는 부족한 면이 있다. 추론의 경우에도 표현된 지식에 국한된 연역적 추론만을 지원하는 한계점을 가지고 있다. 이는 인간의 추론 방식 중 상당히 제한적인 추론으로, 향후 유비추론과 같은 고등의 추론에 대한 지원이 이루어져야 한다. 따라서 본 연구팀은 편리한 온톨로지 편집을 위한 방법적 측면과 온톨로지 통합 및 추론에 대해 설명한다.

2. 온톨로지 관리 프레임워크

온톨로지 관리도구가 갖추어야 하는 모듈 또는 기능에는 많은 것들이 있을 수 있다. 그러나 모든 사용자의 요구를 충족시켜 주는 도구를 만든다는 것은 현실적으로 불가능하며, 또한 비효율적인 작업이 될 수 있다. 따라서 본 연구팀은 온톨로지 관리 프레임워크로써 온톨로지 편집, 통합 그리고 추론과 관련된 부분에 초점을 맞추고, 그들이 갖추어야 할 기능들을 구상하고 구현하는데 중점을 두었다.

2.1 온톨로지 에디팅

온톨로지 편집은 온톨로지 내의 개념, 속성들을 추가 또는 삭제하거나 수정하는 과정 모두를 포함한다. 따라서 온톨로지 편집을 위한 도구들은 편집과정에서 발생할 수 있는 데이터 소실 및 에러에 대해 데이터 안정성을 보장하고, 편집을 위한 편리한 인터페이스를 지원해야 한다.

1) 다중 편집

방대한 양의 온톨로지를 손쉽게 편집하기 위해서는, 동시에 여러 온톨로지에 대한 편집이 가능하여야 한다. 이를 위해 기본적으로 두 개의 패널을 통해 양측에 대한 에디팅이 가능하며, 추가적으로 다른 패널을 띄워 다중 온톨로지에 대한 편집이 가능하도록 하였다.

2) 텍스트 기반 편집 지원

온톨로지 편집에 있어 가장 복잡한 작업은 Restriction Class를 핸들링 하는 것이다. 대표적으로 protege 프로그램의 Asserted Condition 패널을 그 예로 들 수 있다. Restriction Class에 대한 표현은 DL에 기반하며, 특정 논리 연산자들과 온톨로지 내의 개념, 속성들의 조합으로 표현되어진다. 앞에서 예로 든 protege 프로그램의 경우에도 버튼들을 클릭하고 다이얼로그들을 호출하는 방식으로 논리 연산자들과 개념 또는 속성들의 조합으로 Restriction Class를 생성 또는 편집한다.[3] 그러나 이러한 방식은 표현식 생성에 있어서 버튼 클릭과 같은 번거로운 작업들이 많은 단점이 있다.

이에 반하여 텍스트 기반의 편집을 지원함으로써, 사용자가 직접 키보드를 통해 표현식을 생성할 수 있고 편집을 가능하게 함으로써 불필요한 작업들을 줄이고, 손쉽게 표현식을 다룰 수 있다. 이러한 편집 방식은 다음과 같은 장점을 지닌다.

첫째, 버튼 클릭과 같은 불필요한 작업을 줄일 수 있다.

둘째, 구축된 온톨로지에 구애받지 않고 새로운 클래스 또는 프로퍼티를 표현할 수도 있다.

셋째, 다양한 문법적 표현이 가능하다.

이를 위해 잘못 입력된 표현식에 대한 문법적 정당성 체크 및 새로운 클래스 또는 프로퍼티에 대하 자동 인식, 그리고 다양한 논리 연산자의 지원과 입력된 문법적 표현을 파싱하기 위한 모듈들이 추가 지원된다.

3) 한국어 사용자를 위한 문법 지원

DL 표현식은 일반 사용자 특히 한국어 사용자에게 익숙치 않은

문법 구조를 가지고 있다. 이는 사용자가 클래스에 대한 표현식을 생성 또는 편집하는데 있어 어려움을 가져다 준다. 따라서 DL에 대한 문법 구조를 한국어 사용자에게 익숙한 문법으로 변환 지원함으로써 표현식 생성 및 편집에 용이성을 주고자 하였다.

(DL 구문) $C_1 \exists P_1 (C_2 \cap C_3)$

위와 같은 DL 표현식에 대해 다음과 같은 구문을 변경 지원한다.

(변경 구문) C_1은(는) (C_2를 P_1한다) 그리고 (C_3를 P_1한다)

이렇게 함으로써 좀 더 한국어적 문법에 맞도록 하여, 사용자가 쉽게 이해 할 수 있도록 하였다.

또한 존재 양화사 같은 특수한 논리 연산자 기호의 경우 키보드를 통한 입력이 힘들기 때문에, 논리 연산자를 영어 표현으로 대체 가능하도록 하였으며, 만약 논리 연산자로 지정된 지시어와 동일한 이름을 갖는 클래스를 정의할 필요가 있을 경우, 따옴표("")를 통해 연산자로 사용된 지시어들과 구분 가능토록 하였다.

4) 템플릿 지원

Restriction Class표현에 있어 템플릿 생성의 필요성은 다음과 같다.

- Restriction Class는 복잡한 표현식으로 구성되는 경우가 많다. 따라서 매번 Restriction Class를 생성할 때마다 새로 작성하는 것은 시간 소모적인 작업이다.
- Restriction Class는 DL에 따라 기술되어지므로, 표현 형식에 있어 일정한 패턴을 지닌다.

이에 사용자는 자주 사용되는 표현이나, 원하는 표현에 대해 템플릿을 생성할 수 있으며, 필요한 때에 한해 저장된 템플릿을 호출하여 사용할 수 있다. 템플릿 생성에 대한 방법은 특정 표현에 들어갈 논리 연산자 또는 클래스, 프로퍼티들에 대한 정보를 미리 지정하게 되는데, 예를 들어 $A \exists P_1 (B \cap C)$과 같은 표현식에 대해 A는 온톨로지에서 A의 하위 클래스만을 입력할 수 있도록 하고, B 또는 C에 대해서는 온톨로지 내의 B, C에 관계된 클래스만을 입력 가능하도록, A, B, C에 대한 정보만을 표현식에 추가되면, A, B, C는 템플릿에 저장될 때 버튼으로 저장된다. 따라서 나중에 템플릿이 호출되었을 경우 A 위치의 버튼을 클릭하면 A의 하위 클래스에 관한 트리가 다이얼로그 박스 형태로 나타나게 되어 선택 입력된다. 이러한 템플릿은 개발된 관리도구 내에 구현된 룰에 따라 동작하게 된다.

5) Semi-Automatic 편집

기존의 온톨로지 관리도구는 클래스 생성에 있어 편집하는 온톨로지에 제한되어 있다. 따라서 클래스에 대한 표현식은 이미 구축된 온톨로지 내의 클래스 및 프로퍼티의 조합만이 가능하다. 그러나 텍스트 기반의 편집을 지원함으로써 새로운 클래스 또는 프로퍼티에 대한 표현이 가능하기 때문에 이를 위해 사용자로부터 입력된 표현식 내에 포함되어진 클래스와 프로퍼티를 자동 인식할 수 있는 모듈이 요구되었다. 표현식 내의 위치에 따라 클래스와 프로퍼티가 지정됨으로 이를 인식하여, 새로운 것일 경우 기존에 구축된 온톨로지에 새로 정의하게 된다.

(예1) $C_2 \exists P_1 C_1$ ($C_1, C_2 \Rightarrow O_1$, $O_1 \not\Rightarrow P_1$)

위 예1과 같은 입력된 표현식에 대해, P_1은 자동적으로 C_2와 C_1을 각각 도메인과 레인지로 정의되어 온톨로지 내에 저장된다.

(예2) $C_2 P_1 C_1$ $C_3 \not\subset C_2$ $C_4 \not\subset C_1$

$C_1 \sim C_4$는 기존 온톨로지 존재하며, 다음과 같은 표현식에 대해 새로운 프로퍼티 P_1에 대해 사용자와의 피드백을 통해 도메인과 레인지를 다시 설정할 것인지를 묻게 된다.

2.2 온톨로지 통합

여러 온톨로지를 통합하는 작업은 온톨로지 내의 개념들 사이의 유사성을 바탕으로 통합이 이루어진다. 그러나 그 의미와 작업의 복잡성으로 인해, 그동안 많은 연구와 다양한 통합 방법들이 제시되어왔다. 일반적으로 통합 작업은 크게 개념들 사이의 어휘적 매핑과 구조적 매핑에 의해 유도된 매핑 쌍들을 바탕으로 통합하는 방식을 취하고 있다.

1) 어휘적 매핑

어휘적 매핑이란 개념들이 갖는 어휘(문자열)를 기반으로, 개념들 사이의 상호 문자열의 유사성을 측정하여 매핑하는 것을 말한다. 본 프로그램에서는 개념들 사이의 어휘적 유사성을 측정하는 두 개의 알고리즘을 도입하였다.

i) SM Algorithm[4]

$$SM(T_i, T_j) = \max\left(0, \frac{Min(|T_i|,|T_j| - ED(T_i, T_j))}{Min(|T_i|,|T_j|)}\right) \in [0,1]$$

위 알고리즘은 Maedche e staab가 2001년 제안한 SM(similarity Measure) 알고리즘으로 개념들의 이름에 대한 문자열(Term)의 최소 길이와 비교되는 두 문자열 중 상호 일치하지 않는 문자의 길이를 바탕으로 유사성을 측정하게 된다.

ii) N-gram Algorithm

$$S = \frac{2Cwq}{Aw + Bq}$$

N-gram이란 문자열 중의 N character slice를 의미하며, 문자열들을 N-gram으로 나누어서 상호 비교하는 것을 의미한다. 보통의 경우 문자열을 두 개의 문자 단위로 N-gram을 구성하며, 두 비교되는 문자열의 N-gram들 사이의 공통된 개수를 바탕으로 유사성 측정이 이루어진다.

2) 구조적 매핑

개념들의 구조적 유사성을 판별하기 위한 초기값으로 보통 어휘적 유사성 측정값을 사용한다. 본 연구팀은 Similarity Flooding Algorithm(SFA)[5]을 사용하여 구조적 매핑을 수행하였다. SFA에서 그래프 상의 모든 노드들에 대해, 매핑쌍들이 생성되며, 각 매핑쌍에 대한 유사성이 측정된다. 매핑쌍들에 대한 측정된 유사성은 그 이웃하는 노드들로 전달되어지는데, 이는 높은 유사성을 갖는 노드들은 그 이웃하는 노드들의 유사성 역시 유사할 가능성이 많을 것이라는 직관에 기반한다.

3) 통합 과정

통합 과정에서 사용자는 통합에 사용되는 알고리즘과 각 알고리즘의 파라메터 값을 선택 지정할 수 있으며, 매핑이 수행되는 깊이를 지정할 수도 있다. 그리고 두 온톨로지 내의 개념들 사이의 관계를 명시적으로 지정할 수 있다. 예를 들어, 사용자는 통합을 수행하기 전, 온톨로지 상의 상호 개념들의 관계로 Equivalent, Sub 또는 Super 관계를 지정해 줄 수 있다. 이러한 사용자의 선택을 바탕으로 순차적으로 어휘적 매

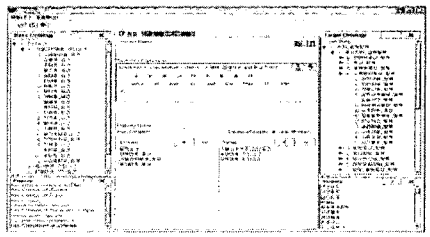
핑과 구조적 매핑 작업을 통해 온톨로지를 통합한다.

2.3 온톨로지 추론

온톨로지 추론 모듈은 편집된 온톨로지에 대한 구문의 정당성 체크및 향후 유비추론을 위한 프레임워크를 위해 관리도구에 포함되어졌다. 현재는 기존의 개발된 온톨로지 추론 엔진(pellet[6])을 연동하여 추론 기능이 동작되도록 하였고, 이는 온톨로지 편집 작업 후 모순되는 부분을 검출하고, 편집된 온톨로지에서 발생할 수 있는 구문에 대한 정당성(Validation)을 체크하거나 편집된 온톨로지 내의 개념들 사이의 관계, 예를 들면, Equivalent 또는 계층적 관계를 추론해 내는데 사용되었다. 그리고 RDQL을 사용 쿼리를 작성하여 특정 정보를 추론해 내는데 사용되었다.

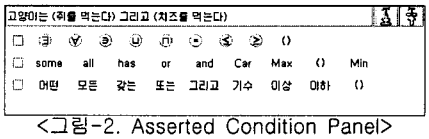
3. 결과

온톨로지 관리도구가 갖추어야 할 기능들과 모듈들에 대한 아이디어를 바탕으로 관리 툴이 구축되었다. 온톨로지에 대한 편집과 통합, 추론 기능이 추가되었으며, 그 구현은 초보적 단계에서 이루어졌다.



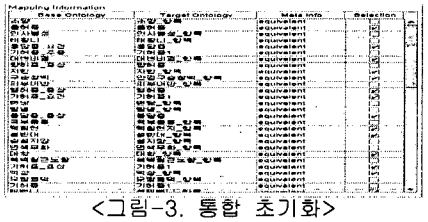
<그림-1. 인터페이스>

그림-1은 구현된 관리도구의 기본 인터페이스를 보여주고 있다. 양측 패널의 온톨로지로부터 구성된 트리부분과 가운데 패널은 온톨로지를 편집을 위한 것이다.



<그림-2. Asserted Condition Panel>

그림-2는 Restriction Class 표현식을 생성하는 부분으로 표현식에 사용되는 여러 논리 연산자들을 선택 사용할 수 있게 하였고, 사용자가 직접 표현식을 서술할 수 있는 패널을 보여주고 있다.



<그림-3. 통합 초기화>

그림-3은 온톨로지들을 통합하는데 있어 초기화된 값들을 보여주고 있다. 실제적인 통합 작업은 초기화된 값을 바탕으로 온톨로지 내의 개념들의 구조적 유사성을 측정해 나가면서 통합이 이루어지게 된다.

4. 결론 및 고찰

몇 해 사이에 지식표현 방법으로써의 온톨로지에 대한 관심이 많아졌다. 이로 인해 온톨로지를 구축하고, 구축된 온톨로지에 대한 효율적 관리 측면 또한 그 중요성이 높아지고 있다. 본 연구팀은 이번 연구를 통해, 한국어 사용자를 대상으로 한 온톨로지 관리도구가 가져야 하는 기본 프레임워크와 기능들에 대해 고려하였다. 물론 본 연구팀이 제안한 방식이 한국어 사용자를 위한 온톨로지 관리도구가 가져야 하는 전부는 아니다. 앞으로의 계속적인 연구를 통해 좀 더 체계적이고, 효율적인 방식이 도입되어야 할 것이다.

5. 향후 과제

향후 본 연구팀은 아래와 같은 기능들을 구상하고 구현하는데 중점을 두고 있다.

1) Intermediate Interface 추가
온톨로지 구축 및 편집에서 클래스 표현식이 복잡할 경우 편집과정에서 중간 인터페이스를 두어, 복잡한 표현식을 손쉽게 생성 가능하도록 하였다. 이는 후에 본 관리 도구에 포함되어질 예정이다.

2) Multiple User Editing and Online Editing
온라인을 기반으로 다중 사용자들이 온톨로지를 편집할 수 있는 기능 구현에 목적이 있다.

3) Analogical Reasoning
온톨로지에 대한 추론에 대해 유비추론을 지원할 수 있는 기능을 포함할 것이다. 이 기능은 현 관리도구의 프레임워크에 기반하여 구현되어질 예정이다.

4) Class-Instance Integration
상호 통합되는 온톨로지 사이의 레벨이 다르거나 문맥(Context)가 다를 경우에 있어, 통합을 지원하기 위한 모듈이며, 관리도구의 통합 프레임워크에 기반하여 구현되어질 것이다.

5) RDB와 Ontology 상호 변환
온톨로지를 데이터베이스와 연동하여 대규모의 온톨로지를 효율적으로 관리하기 위한 모듈이다.

6) Ontology Query Algorithm
온톨로지에 대한 유비추론을 수행하기 위한 쿼리에 대한 알고리즘을 개발하여 향후에 구현되는 유비추론 모듈과 더불어 실제적인 온톨로지에 대한 유비추론을 수행하게 된다.

6. 감사의 글

This study was supported by a grant of the Korea Health 21 R&D Project, Ministry of Health & Welfare.
(03-PJ1-PG10-51300-0003)

7. 참고 문헌

1. http://protege.stanford.edu/
2. Sean Bechhofer, Lan Horrocks OilEd: a Reason- able Ontology Editor for the Semantic Web. Proceedings of KI2001, Joint German/Austrian conference on Artificial Intelligence, September19-21, Vienna. Springer-Verlag LNAI Vol. 2174, pp 396-408.
3. Matthew Horridge, A Practical Guide To Building OWL Ontologies With The Protege-OWL Plugin Edition 1.0
4. Alexander Maedche and Steffen Staab, Comparing Ontologies-Similarity Measures and a Comparison Study, Institute AIFB, University of Karlsruhe Internal Report 2001
5. Sergey Melnik and Erhard Rahm Similarity Flooding: A Veersatile Graph matching Algorithm and its Application to Schema Matching Published in Proc. 18th Intl. Conf. on Data Engineering(ICDE), San Jose CA, 2002
6. http://www.mindswap.org/2003/pellet/index.shtm