EOL : SUNHI 표현범위를 가진 인식론적 온톨로지 표현 언어 와 추론엔진

EOL : Epistemological Ontology Language and Reasoner with SUNHI for Ubiquitous Computing Environment

마종수, Jongsoo Ma*, 김민수, Minsoo Kim*, 김민구, Minkoo Kim**
*아주대학교 정보통신전문대학원, **아주대학교 정보및컴퓨터공학부

요약 ˇ ˇ현재 이슈가 되고 있는 유비쿼터스 컴퓨팅 환경에서 서비스를 제공함에 있어 사용자의 만족도를 높여주기 위해 서비스의 지능화가 필요하다. 이러한 지능적인 서비스를 제공하기 위해 서비스에 필요한 지식을 논리적으로 표현하고, 체계적으로 추론할 수 있는 방법이 요구된다. 이를 위해 표현 범위가 넓고 유연한 일차 술어 논리(FOL)는 여러 분야에서 사용되었으며, 추론 시스템에 이용되고 있다. 그러나 풍부한 표현 범위는 유비쿼터스 컴퓨팅 환경에서의 오브젝트 관리에 있어 많은 계산비용이 소요된다. 서비스의 빠른 제공을 목표로 하고 있는 유비쿼터스 환경에서 이러한 계산비용은 서비스 제공 시간을 늦추는 요인이 된다. 이러한 문제를 극복하고 지식의 의미를 부여하는 방법으로 Description Logic과 온톨로지가 연구되고 있다. 특히 OWL(Web Ontology Language)은 풍부한 표현력을 제공하고 있으며, W3C에 의해 온톨로지 기술의 표준으로 제안되었다. 그러나 풍부한 표현 범위는 실제 컴퓨팅 환경에서 모두 사용되지 않고, 기술 및 추론의 복잡함으로 overhead가 발생한다. 본 논문에서는 이를 극복하고자 실제 유비쿼터스 환경에서 요구되는 표현 범위를 만족하는 SUNHI의 표현력을 갖는 EOL을 제안한다.

핵심어: 서술논리, 유비쿼터스컴퓨팅

1. 서론

유비쿼터스 컴퓨팅 환경에서의 서비스는 사용자의 명령없이 오브젝트 스스로 수행되는 자율적인 서비스를 기대하고 있다.[1] 이러한 자율적인 서비스를 제공하기 위해서는 각 서비스 오브젝트들의 지능적인 동작이 보장되어야 한다. 이는 서비스를 수행케 하는 지식의 표현과 추론기능이 연동됨을 의미한다.

지식을 표현하고, 이 지식을 바탕으로 새로운 지식을 추론하고, 활용하는 연구가 계속 진행되어 왔다. 이들 방법으로는 수리논리(Mathematical Logic), 의미망(Semantic Logic), 프레임(Frame), 생성 규칙(Production Rules) 및 여러 방법의 조합인 합성(Hybrid) 등이 있다. 이런 지식표현 방법 중 수리 논리를 기반으로 한 일차 술어 논리 기반 표현 방법(First Order Predicate Calculus)이 지시 기반 시스템을 구현하는데 많이 사용되어왔다. FOPC 기반 지식처리방법은 지식 표현의 한계가 없으며, 표현된 지식의 추론 결과 증명이 용이한 장점을 가지고 있다. 하지만 추론 과정이 복잡하고, 계산비용이 크다는 단점을 동시에 가지고 있다. 이러한 단점을 보완하기 위해 Description Logic을 기반으로 온톨로지를 사용하는 방법이 제안되었다.

본 연구는 21세기 프론티어 연구개발사업의 일환으로 추진되고 있는 정보통신부의 유비쿼터스컴퓨팅및네트워크원천기반기술개발사업의 지원에 의한 것임

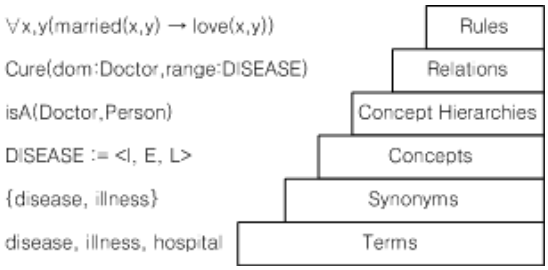
2. 관련연구

2.1 Description Logic 과 온톨로지

온톨로지는 공유되는 개념에 대한 정형화되고 명백한 명세서로 정의된다.[2] 공유되는 개념이란 동일한 환경에 존재하는 다양한 시스템들이 지능적인 동작을 수행하기 위해 작성한 환경 모델을 서로 이해할 수 있다는 것이다. 또한 이러한 모델 기술을 위한 컨셉, 관계, 제약조건 등의 문법을 통해 명시적으로 선언되어져야 한다. 온톨로지 표현은 다음과

같은 구조로 이루어진다<그림1>[3]. 이처럼, 지식 공유를 위한 수단으로 온톨로지가 사용됨에 따라 온톨로지를 표현하는 방법이 연구되었다. 온톨로지를 표현하는 언어로써 RDF, OIL, DAML, DAML+OIL, OWL 등이 존재한다.

그림 1 ontology layer cake



Description Logic은 Frame 기반으로 대상을 기술한다. 이 대상으로는 술어지식(Terminological Knowledge Base : TBox)과 정언지식(Assertional Knowledge Base : ABox)로 나눌 수 있다. TBox는 개념들과 그 속성, 그리고 개념들과의 관계로 지식을 표현한다. ABox는 TBox에 정의된 개념들의 개체로 지식을 표현하며, ABox의 개체들은 각 도메인에 밀접한 관련을 가지고 있다. 각각의 지식이나 개념의 표현에 있어서 Description Logic은 지식을 서술하는 논리식이 정형화 되어 있다. 또한 표현 정도에 따라 Description Logic을 구분할 수 있는데 <표 1>은 Description Logic의 단계를 보여준다.

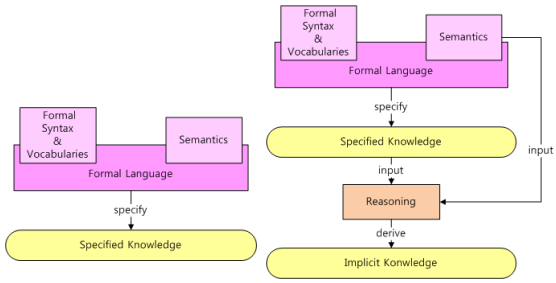
표 1 Description Logic Family

DL Family		expressive power	semantics
S	AL	A	atomic concept
		⊤	universal concept
		⊥	bottom concept
		¬A	atomic negation
		C⊓D	intersection
		∀R.C	value restriction
		∃R.⊤	limited existential quantification
	C		negation of arbitrary concepts
	R+		transitive roles
U		C⊔D	union
E			full existential quantification
N		≥nR, ≤nR	number restriction
H			role hierarchy
I		R^{-}	inverse role
Q		≥nR.C, ≤nR.C	qualified number restriction
O		|I| = 1	nominal

2.2 추론 알고리즘

추론이란 어떠한 판단을 근거로 삼아 다른 판단을 이끌어내는 것을 의미한다. 단순한 가정이 아닌 사실을 근거로 하여 결론을 유도해내는 것을 말한다. 이러한 추론은 형식 언어를 이용하여 명시적으로 기술된 명시 지식과 명시적으로 기술되지 않았으나, 암묵적으로 참인 지식인 암묵 지식이 있다<그림 2>[4].

그림 2 명시지식 & 암묵지식



추론 알고리즘으로는 TBox만을 이용한 구조적 포함 알고리즘이 있고, TBox와 ABox를 모두 사용하는 Tableau 알고리즘이 있다.

구조적 포함 알고리즘은 TBox에 정의된 개념을 가지고 어떤 새로운 개념의 만족여부 혹은 두 개념 사이의 포함관계를 알아내는데 쓰인다. TBox에는 기초 개념과 복합 개념이 존재한다. 이 알고리즘은 sound 하고 complete하지만, 추론 표현력의 한계로 인해 낮은 수준의 표현 밖에 할 수 없다.

Tableau 알고리즘은 직접 개념의 구조를 비교하는 대신, 간접적으로 포함관계를 추론한다. 포함관계 문제는 만족 여부 문제로 바꾸어서 풀 수 있다. Tableau 알고리즘의 핵심은 TBox와 ABox를 이용하여 tableau가 생성되는지를 풀어내는데 있다. 여기서 tableau가 생성된다는 말은 개념과 개체 모두 존재한다는 것이다. Tableau 알고리즘은 논리와 개체비교를 통한 간접적 추론을 하기 때문에 논리로 풀어낼 수 있는 높은 표현력의 추론도 가능하다.

3. EOL 로의 확장

Description Logic은 TBox와 ABox를 이용하여 환경 모델을 만든다. TBox의 공유를 통해 지식 모델이 공유될 수 있고 FOPC 기반 지식보다 활용 및 계산이 보다 효율적이다. 하지만 추론방법으로 주로 사용되는 Tableau-based 알고리즘의 경우, SHIQ의 표현범위로 증가하면 지수복잡도로 증가한다[5].

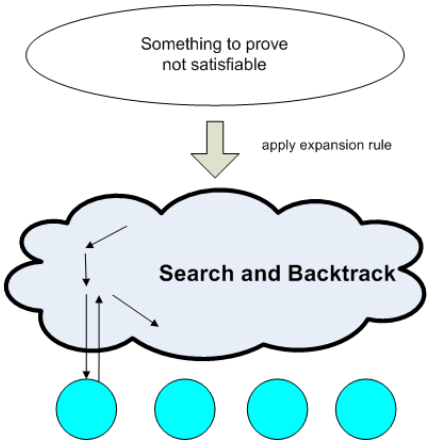
일반적으로 유비쿼터스 컴퓨팅 환경에서 지능화된 서비스를 받기 위해서는 existential quantification, value restriction, number restriction, intersection, union, transitive role, inverse role, number restriction 및 role hierarchy의 지식 표현 능력을 갖고 있어야 한다. 이러

한 SUNHI의 표현범위를 갖은 EOL은 사용자 직관적인 명령어와 표현문법을 제공해 지식을 활용하여 지능적인 시스템 구축을 도와준다. OWL의 OWL-Lite는 Sound & Completeness 가 증명되었지만, OWL-DL은 아직도 완벽하게 지원되는 추론 엔진이 없고, OWL-FULL을 완벽히 지원하기는 현실적으로 힘든 상황이다 [6].

3. 실험 및 평가

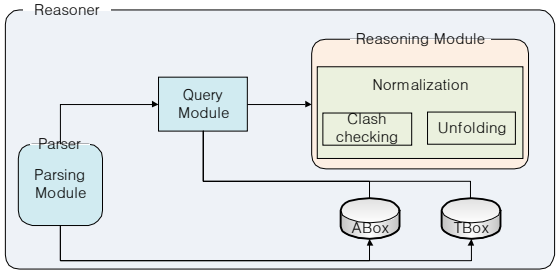
EOL의 추론 알고리즘은 많은 추론엔지에서 사용되고 있는 tableau-based 알고리즘을 사용하였다. 이 추론 알고리즘은 negation에 대해서 다양한 변환 규칙을 적용하여 satisfiable 하지 않다는 것을 보여주는 것이다<그림 3>.

그림 3 tableau 알고리즘 개념도



먼저 SUNHI의 표현력을 갖는 EOL의 기본 구조는 다음과 같다<그림 4>.

그림 4 EOL 기본구조도



Reasoner는 Parsing Module, Query Module, 그리고 Reasoning Module로 크게 세부분으로 나눌 수 있다.이 부분은 분리해서 따로 설명하도록 하겠다.

3.1 EOL의 문장구조

EOL의 syntax는 RACER Pro의 syntax를 기본으로 작성하였다. <표 2>는 EOL의 BNF 문법을 보여준다.

표 2 EOL의 BNF Syntax

```
<concept> ::= (0+ <composite_concept>
<restriction>0+ )0+

        | (0+ <concept_name> <restriction>0+ )0+
<composite_concept> ::=
<intersection_of_concepts>| <union_of_concepts>
                    | <negation_of_concept>
<restriction> ::= <universal_restriction>
| <existential_restriction>
| <at_most_restriction>| <at_least_restriction>
<universal_restriction> ::= (all <role_name>
<concept>)
<existential_restriction> ::= (some
<role_name> <concept>)
<at_most_restriction> ::=
                    (at_most <digit> <role_name>)
<at_least_restriction> ::=
                    (at_least <digit> <role_name>)
<negation_of_concept> ::= (not <concept>)
<intersection_of_concepts> ::=
                        (and <concept> <concept>)
<union_of_concepts> ::= (or <concept>
<concept>)
<concept_name> ::=
<uppercase_letter><letter_or_digit>0+
<role_name> ::=
<lowercase_letter><letter_or_digit>0+
<Maximal_concept> ::= *TOP*
<letter_or_digit> ::= <letter> | <digit>
<letter> ::=
<lowercase_letter>|<uppercase_letter>
<lowercase_letter> ::= a|b|c|…|y|z
<uppercase_letter> ::= A|B|C|…|Y|Z
<digit> ::= 0|1|2|3|…|9
```

이러한 문법은 <표 3>, <표 4>의 명령어를 통해 조합이 가능하며, 이는 추론 및 활용을 할 수 있게 해준다. .

표 3 Description Logic Family

Commands for defining concept	
DefineConcept	(defineconcept <concept_name> <concept>)
Concept subsumption	(implies <concept_name1> <concept_name2>)
Concept Disjointness	(disjoint <concept_name1> <concept_name2>)
Commands for defining role	
Define Role	(definerole <role_name>.<concept1> <concept2>)
Role transitive	(transitiverole <role_name>)

Inverse Role	(inverse <role_name> to <role_name>)
Role Hierarchy	(subrole <role_name> of <role_name>)
Commands for asserting instances	
Asserting Instance	(instance <concept> <instance_name>)
Commands for querying	
Unsatisfiability check	(Isunsatisfiable <concept>)
Equivalence check	(Isequivalent <concept1> <concept2>)
Subsumption check	(Issubsume <concept1> <concept2>)
Disjointness check	(Isdisjoint <concept1> <concept2>)
Superconcepts retrieve	(superconcepts <concept>)
Role lists retrieve	(roles <concept>)
Instance retrieve	(instance <concept>)
Role relation retrieve	(role <condition> <role_name>?)

표 4 knowledge base 구축을 위한 추론 엔진 명령 예제

```
(defineConcept Animal)
(defineConcept Male)
(defineConcept Female (and Animal (not Male)))
(defineConcept Person)
(defineConcept Woman (and Person (not Male)))
(defineConcept Man (and Person (Male)))
(definerole hasFather.Person Person)
(definerole hasMother.Person Person)
```

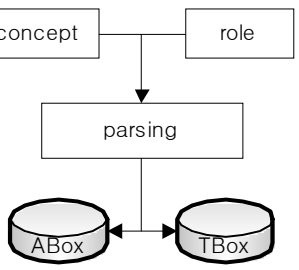
3.2 EOL의 구조

EOL의 각 module들의 기능을 분석하고 그 구조도를 살펴보고자 한다.

3.2.1. Parsing Module

Parsing Module은 말 그대로 기존의 knowledge base의 loading 이나 새로운 knowledge base 의 생성 및 저장을 담당한다. 이렇게 저장되어 있는 knowledge는 프로그램이 구동되는 중 선택되어 메모리상에 올라오게 된다.

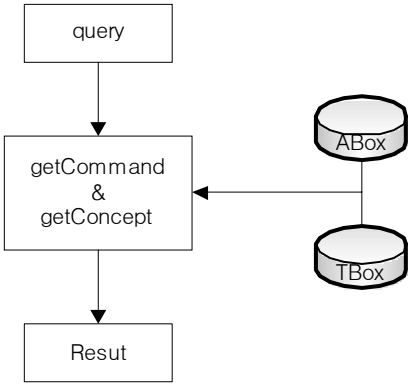
그림 5 Parsing Module



3.2.1. Query Module

추론을 하기위한 쿼리문을 분석하는 부분이다. 쿼리로 들어오는 부분을 명령어와 concept, role의 관계를 파악하고 분류하여 reasoning module에 넘겨주게 된다. 이 부분은 쿼리를 정확히 작성하였는지를 검사한다. 잘못된 쿼리문이 reasoner에서 처리하는 것 보다 단순 string을 처리하는 것이 간편하고 필요없는 계산을 줄일 수 있게 된다.

그림 5 Parsing Module



3.2.1. Reasoning Module

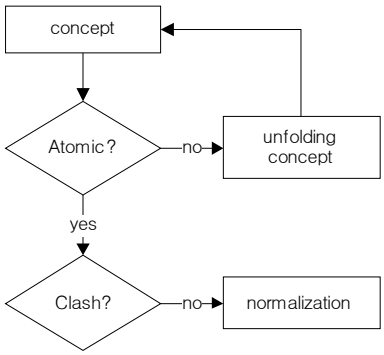
Reasoning Module은 knowledge base를 읽어들인 후, 사용자의 입력을 통해 knowledge base 확장 혹은 reasoning의 역할을 수행한다.

여기서 normalization 을 해주어야 하는데 이는 추론을 위한 지식의 전처리라고 볼 수 있다. 현재 사용자의 입력을 knowledge base에 추가할 것인지, 추론을 위한 것인지를 unfolding과 clash checking를 통해 알 수 있다.

unfolding은 쿼리로 들어온 사용자 입력에 대해 단일 개념의 컨셉의 논리곱으로 표현하게 한다. 이는 컨셉 간의 포함관계 등을 계산하기 위해 필요한 작업이다.

unfolding이 끝나면 이 자체에 대한 clash checking을 하게 된다. 이는 unfolding을 통해 만들어진 트리구조의 컨셉, 롤 계층구조를 통하여 충돌유무를 확인한다.

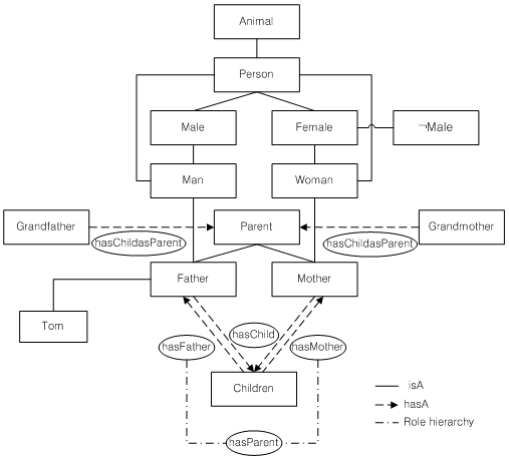
그림 6 Normalization Module



3.3 EOL의 적용 사례

이제 사례를 통해 EOL의 적용사례에 대해 이야기하고자 한다. <그림 7>은 간단한 가계도를 나타난 것이다.

그림 7 가족관계 ontology



이 관계를 EOL을 통하여 변환하게 되면 <표 5>와 같은 결과를 얻게 된다. 이러한 표현결과는 기존의 protege 등 에디터의 도움없이 사용자가 직접 타이핑을 통해 지식 입력이 가능하고, 가독성 또한 높여준다.

표 5 EOL을 이용한 가계도 정의

```
(defineConcept Animal)
(defineConcept Male)
(defineConcept Female (and Animal (not Male)))
(defineConcept Person)
(defineConcept Woman (and Person (not Male)))
(defineConcept Man (and Person (Male)))
(definerole hasFather.Person Person)
(definerole hasMother.Person Person)
(definerole hasParent (and hasFather hasMother))
(definerole hasChild.Person Person)
(defineConcept Mother (Woman (some hasChild)))
(defineConcept Father (Man (some hasChild)))
(defineConcept Parent (or Mother Father))
(defineRole hasChildasParent.Person Parent)
(defineConcept GrandMother (and Woman (some haschildasparent Person)))
(defineConcept GrandFather (and Man (some haschildasparent Person)))
(implies Animal Person)
(implies Person Man)
(implies Person Woman)
(implies Parent Father)
(implies Parent Mother)
(disjoint Man Woman)
(inverse hasChild to hasParent)
(subrole hasFather of hasParent)
(subrole hasMother of hasParent)
(instance Father Tom)
```

제시된 가족 온톨로지를 대상으로 EOL의 추론과정을 설명하고자 한다.

- (Isunsatisfiable Mother)

Mother가 컨셉으로써 만족한지 알아보자. Mother는 hasChild를 갖는 Person이면서 Man이 아니다. Man은 Person 이면서 Male 이므로, Mother는 hasChild를 갖는 Person 이면서 Male이 아니다. Person은 Animal에 포함관계를 갖게 됨으로 원자개념 까지 내려갈 수 있으므로 satisfiable 하다. 그러므로 거짓을 반환한다.

- (Isequivalent Father Tom)

Tom의 컨셉이 Father 인지를 알아보는 쿼리문이다. (instance Father Tom) 에서의 선언과 같이 Tom은 Father이기 때문에 참을 반환한다.

- (Issubsume Mother Male)

컨셉들간의 포함관계 중 subsume관계에 대한 쿼리문이다. Mother은 Parent 이면서 Male이 아니다. 그러므로 Male의 subsume이 될 수 없기 때문에 거짓을 반환한다.

- (Isdisjoint Father Mother)

컨셉들간의 포함관계 중 disjoint에 대한 쿼리문이다. Father와 Mother는 모두 Person의 속성을 가지고 있기 때문에 disjoint하지 않다. 그러므로 거짓을 반환한다.

- (roles hasParent)

Role의 계층관계를 나타내는 부분이다. hasParent는 hasFather과 hasMother의 union으로 구성되어 있으므로 hasFather과 hasMother이 반환된다.

- (instance Tom)

instance 뒤에 두개의 컨셉이 오면 컨셉을 갖는 instance가 형성되는 것이지만, 한 개의 컨셉이 오면, 그 instance의 컨셉을 물어보는 쿼리문이 된다. Tom은 Father의 컨셉을 가지고 있으므로 Father를 반환한다.

- (role down hasParent)

Role의 계층구조를 보여주는 쿼리문이다. hasParent의 아래 방향, 즉 하위 role을 검색하는 것이다. hasFather 과 hasMother 를 반환한다.

4. 결론

오늘날의 유비쿼터스 컴퓨팅 환경의 대두와 함께 지능적인 시스템에 대한 요구도 증가하고 있다. 보다 지능적인 시스템을 만들기 위해 그 시스템이 사용되는 상황에 대한 인식을 통해 오브젝트들이 보다 상대적으로 적은 컴퓨팅 능력으로 서비스를 제공할 수 있어야 한다. 이러한 이유로 FOPC 같은 고수준의 컴퓨팅 능력이 아닌 Description Logic 기반의 온톨로지를 필요한 단계들을 뽑아내 사용할 수 있어야 한다. 본 논문에서 제안한 SUNHI의 표현능력을 가진 온톨로지 표현 언어인 EOL은 유비쿼터스 컴퓨팅 환경에서 적은 리소스와 계산량으로 지능형 시스템을 구축하는데 활용될 수 있다.

참고문헌

[1] Weiser, M., The Computer for the Twenty-First Century. Scientific American, Sept. 1991, pp. 94-110

[2] Pim Borst, Hans Akkemans, An Ontology Approach to Product Disassembly EKAW97 Sant Feliu de Guixols, Spain October 15-19th 1997

[3] Paul Buitelaar , Philipp Cimiano and Bernardo Magnini, Ontology Learning from Text: An Overview, IOS Press, 2003

[4] 장민수, OWL 추론과 추론시스템 소개, ETRI 2005

[5] Horrocks, I. U. Sattler, and S. Tobies, Reasoning with Individuals for the Description Logic SHIQ proc. of the 13th Conf. on Automated Deduction(CADE-17)

[6] I. Horrocks. The FaCT system. In H. de Swart, editor, Automated Reasoning with Analytic Tableaux and Related Methods: International Conference Tableaux'98, number 1397 in Lecture Notes in Artificial Intelligence, pages 307?-312. Springer-Verlag, Berlin, May 1998.

[7] 조성원, 이건수, 송세헌, 김민구 ALKETge : 유비쿼터스 환경을 위한 지식표현 언어, 2005 정보과학회 추계학술대회

[8] Franz Baader, Ulrike Sattler, An Ontology of Tableau Algorithms for Description Logics, Kluwer Academic Publishers. 2001