OWL 추론과 추론시스템 소개

@KISTI, 2005-09-29
장민수 (minsu@etri.re.kr), ETRI
http://mknows.etri.re.kr

목차

추론(Inference)

정의

1a. The act or process of *deriving logical conclusions from premises* known or assumed to be true. b. The act of reasoning from factual knowledge or evidence.

2a. Something inferred. b. Usage Problem A hint or suggestion: The editorial contained an inference of foul play in the awarding of the contract.

[from The American Heritage® Dictionary of the English Language: Fourth Edition. 2000.]

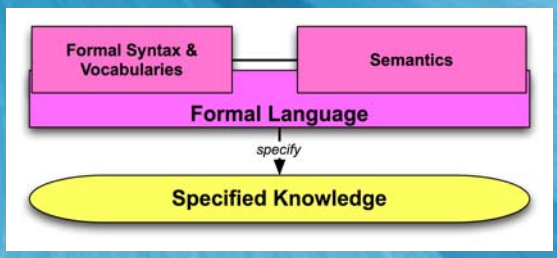
추론(Inference)

명시 지식(Explicit Knowledge) (1)

- 형식 언어를 이용하여 명시적으로 기술된 지식
  - 형식 언어는 지식을 표현할 수 있는 형식 문법(Formal Syntax)과 어휘(Vocabulary)를 제공한다.
  - 각 어휘와 구문 구조는 의미(Semantics)를 내포한다.
  - 형식 언어의 어휘를 이용하여 그 언어의 문법에 따라 기술한 지식

추론(Inference)

명시 지식(Explicit Knowledge) (2)



추론(Inference)

명시 지식(Explicit Knowledge) (3)

- 형식 문법과 어휘
  - X *implies* Y.
  - *fact* X.
- 의미
  - X *implies* Y. : *만약 X가 참이면, Y는 참이다.*
- 명시 지식의 예
  - *Father(?x) implies Man(?x).*
  - *fact Father(Cheolsu).*

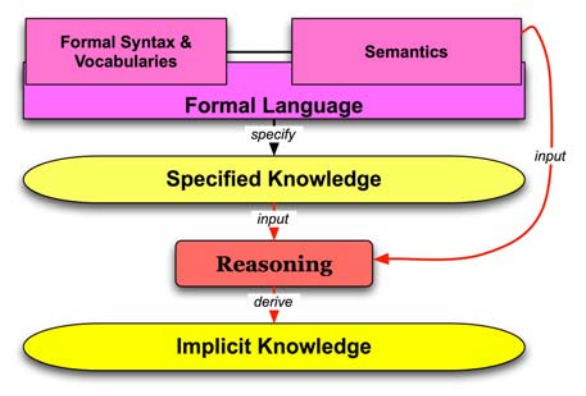
추론(Inference)

암묵 지식(Implicit Knowledge) (1)

- 명시적으로 기술되지 않았으나, 암묵적으로 참인 지식
  - 형식 언어의 의미론(Semantics)과 그 언어로 기술된 명시 지식으로부터 암묵 지식의 진위를 유추할 수 있다.
  - 추론은 명시 지식에 대하여 '추론 규칙'을 반복적으로 적용함으로써 암묵 지식을 유도해내는 과정을 말한다.
  - '추론 규칙'은 형식 언어의 의미론에 의해 정의된다.

추론(Inference)

암묵 지식(Implicit Knowledge) (2)



추론(Inference)

암묵 지식(Implicit Knowledge) (3)

- 추론 규칙
  - X *implies* Y. : {(X -> Y) and X} -> Y *[Modus-Ponens]*
- 추론 규칙의 적용
  - Father(?x) implies Man(?x).
  - fact Father(Cheolsu).

Modus-Ponens를 적용하여 암묵 지식 도출

{(Father(?x) -> Man(?x)) and Father(Cheolsu)}
-> Man(Cheolsu)

추론(Inference)

용도: 일관성(Consistency) 검사

- 주어진 지식이 일관성이 있는가?
  - 주어진 명시 지식을 기반으로 추론을 수행하여 일관성을 해치는 요소가 있는지 판단한다.
- 예) 아래의 지식은 일관성 있는가?

fact Father(Cheolsu).
fact Woman(Cheolsu).
Father(?x) implies Man(?x).
Man(?x) implies ~Woman(?x).
Woman(?x) implies ~ Man(?x).
-->
Inconsistent! *"Man(Cheolsu) and ~Man(Cheolsu)"*

추론(Inference)

용도: 이론(Theory) 검증

- 주어진 명시 지식 하에서, 임의의 이론(Theory)이 타당한지 검증한다.
  - 질의 응답은 이론 검증 기능을 통해 구현된다.
- 예) 다음의 명시 지식이 주어진 경우:

  fact Father(Cheolsu).

  Father(?x) implies Man(?x).

  - 다음의 이론은 타당한가?

  Man(Cheolsu). --> 타당하다.

  - 다음의 이론을 만족시키는 ?x의 값은?

  Man(?x). --> ?x = Cheolsu

추론(Inference)

용도: 유도(Derivation)

- 주어진 명시 지식으로부터 유도될 수 있는 모든 암묵 지식을 생성한다.
- 예) 다음의 지식을 기반으로 유도!

  fact Father(Cheolsu).

  Father(?x) implies Man(?x).

  Man(?x) implies ~Woman(?x).

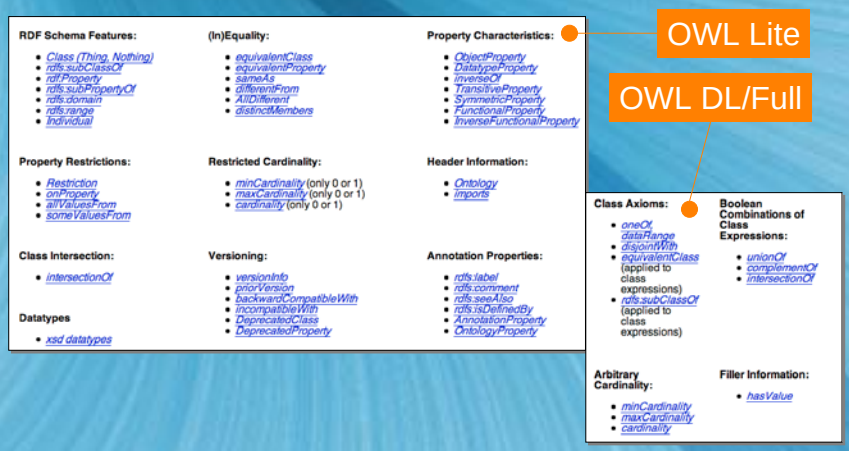
  Woman(?x) implies ~ Man(?x).

  -->

  Man(Cheolsu) and ~Woman(Cheolsu)

OWL 추론

OWL 언어의 구성 어휘



OWL 추론

의미론 예: 클래스 정의 어휘들 (1)

- Class Descriptions & Class Axioms
  - Class Descriptions 아래와 같이 클래스를 기술한다.
    - *Enumerations*
      - owl:oneOf
    - *Boolean compositions*
      - owl:unionOf, owl:complementOf etc
    - *Restrictions*
      - owl:allValuesFrom, owl:cardinality etc
  - Class Axioms 은 클래스 간의 관계를 기술한다.
    - *Subsumption*
      - rdfs:subClassOf
    - *(In)Equality*
      - owl:equivalentClass, owl:differentFrom, etc

OWL 추론

의미론 예: 클래스 정의 어휘들 (2)

- 예1)
  - 구문: Class(C complete unionOf(C1, C2, C3))*
  - 의미: {C} = {C1} ∪ {C2} ∪ {C3}
    - {*c*}: a set of individuals contained in a class *c*
- 예2)
  - 구문: restriction(p hasValue(v))*
  - 의미: {x | p(x,y) ∧ y=v}

** OWL Abstract Syntax. Refer to [3]*

OWL 추론

의미론에 따른 추론 규칙

- OWL의 unionOf
  - 구문: Class(C complete unionOf(C1, C2, C3))
  - 추론 규칙: C1(?i) V C2(?i) V C3(?i) -> C(?i) ---(1)
- 예)

  Class(Human complete
  unionOf(Man, Woman))
  Individual(Cheolsu type(Man))

  -> 추론 규칙 (1)을 적용하여 다음의 결론을 유도.

  Individual(Cheolsu type(Human))

OWL 추론

용도

- Consistency Check
  - OWL 문서의 일관성을 검사한다.
- Subsumption & Equivalence
  - 클래스 간의 계층 관계 및 동치 관계를 알아낸다.
- Membership
  - 개체(Individual)가 어느 클래스에 속하는지 알아낸다.

OWL 추론

Consistency Check 예: 광우병 문제 (1)

```
Class(Cow complete intersectionOf(Animal,Vegetarian))
Class(Sheep partial Animal)
Class(Vegetarian complete
        restriction(eats, allValuesFrom(complementOf(Animal))))
Class(MadCow complete
         intersectionOf(Cow,
                        restriction(eats, someValuesFrom(Sheep))))
```

- Consistent!

  But, MadCow is an unsatisfiable concept!!

OWL 추론

Consistency Check 예: 광우병 문제 (2)

```
Class(Cow complete intersectionOf(Animal,Vegetarian))
Class(Sheep partial Animal)
Class(Vegetarian complete
        restriction(eats, allValuesFrom(complementOf(Animal))))
Class(MadCow complete
         intersectionOf(Cow,
                        restriction(eats, someValuesFrom(Sheep))))
Individual(Cow123 type(MadCow))
```

- Inconsistent!
  The unsatisfiable concept MadCow class has a member!

OWL 추론

Subsumption 예: 가족 구조 (1)

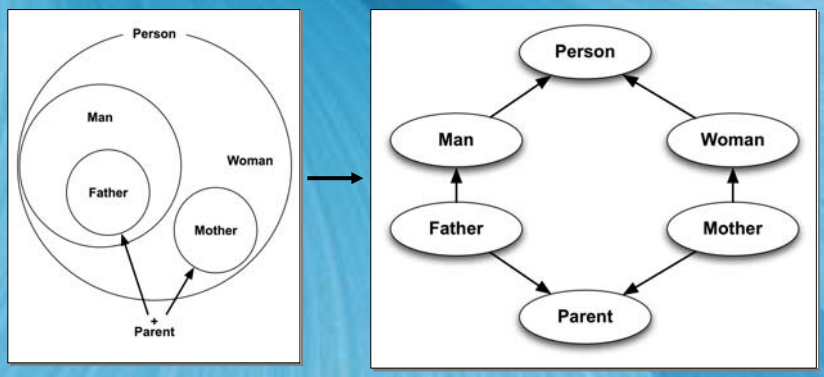
```
Class(Woman complete intersectionOf(Person,Female)))
Class(Man complete
                intersectionOf(Person, complementOf(Woman)))
Class(Mother complete intersectionOf(Woman,
                restriction(hasChild, someValuesFrom(Person))))
Class(Father complete intersectionOf(Man,
                restriction(hasChild, someValuesFrom(Person))))
Class(Parent complete unionOf(Mother, Father))
```

- (X intersectionOf Y, Z) implies (X subClassOf Y and X subClassOf Z)
- (X unionOf Y, Z) implies (Y subClassOf X and Z subClassOf X)

OWL 추론

Subsumption 예: 가족 구조 (2)

추론을 통해 유추된 클래스 계층 구조



OWL 추론

Equivalence 예: Boolean Composition

```
Class(A complete intersectionOf(B C D))
Class(X complete intersectionOf(B D))
EquivalentClasses(C D)
```

- EquivalentClasses(A X)

OWL 추론

Membership 예

```
Class(Vegetarian complete intersectionOf
                    (Animal,
                     restriction(eats, allValuesFrom(Plant))
                     restriction(eats, cardinality(1))))
Individual(Carrot type(Plant))
Individual(Dolly type(Animal) value(eats Carrot))
```

- Individual(Dolly type(Vegerarian))

OWL을 넘어

OWL의 한계: 프로퍼티 조합(Property Composition)

- OWL로 두 개 이상의 서로 다른 프로퍼티를 조합하여 지식을 표현할 수 없다.
  - 그러나, 지식 표현에 있어 필수적임.
- OWL은 다음을 표현할 수 없다.
  - 예1) The *brother* of somebody's *father* is that somebody's *uncle*.
  - 예2) An *employee* of a *member organization* is also a *member*.
- 규칙(Rule)을 이용하면 손쉽게 표현된다.
  - 예1) If father(?x,?y) and brother(?z,?x) then uncle(?z,?y).
  - 예2) If member(?x,?y) and employee(?z,?y) then member(?x,?z).

OWL을 넘어

OWL의 한계: 함수

- OWL로 수식이나 함수 호출을 표현할 수 없다.
  - 그러나, 지식 표현에 있어 필수적임.
- OWL은 다음을 표현할 수 없다.
  - 예1) Teenager is a class of people whose age is greater than 10 and less than 20.
  - 예2) The set of all positive numbers divisible by 3 is ThreeNumber class.
- 함수를 허용하는 규칙을 이용하면 표현 가능
  - 예1) If age(?x,?a) and [?a > 10] and [?a < 20] then Teenager(?x).
  - 예2) If Number(?x) and [mod(?x,3) == 0] then ThreeNumber(?x).

OWL을 넘어

OWL의 한계 : Closed-World Assumption (1)

- OWL은 Open-World Assumption에 기초하고 있다.
  - 어떤 진술이 거짓임은 명시적으로 선언되어 있거나 유추되어야 한다.
  - 즉, 새로운 진술의 등장이 기존에 참인 진술을 거짓으로 만들 수 없다. (단조 추론)
- 반면, Closed-World Assumption은……
  - 만약 어떤 진술이 참이라는 사실이 주어진 지식에 존재하지 않으면, 그 진술은 거짓인 것으로 결론한다. (Negation-As-Failure)
  - 즉, 새로운 진술이 등장하면 이전에 참으로 결론한 진술이 거짓이 될 수 있다. (비단조 추론)

OWL을 넘어

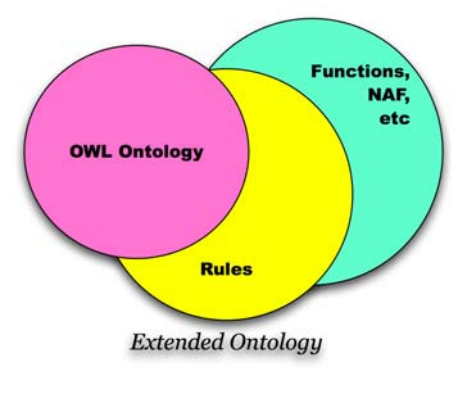
OWL의 한계 : Closed-World Assumption (2)

- KB = {BadGame(a), Game(b), subClassOf(BadGame,Game)}
- OWA)
  - Game(?x) and ¬BadGame(?x) then NiceGame(?x);
  - 결론 없음.
  - KB에 BadGame(b)가 추가됨.
  - 결론 없음.
- CWA)
  - Game(?x) and ~BadGame(?x) then NiceGame(?x);
  - NiceGame(b)를 참으로 결론함.
  - KB에 BadGame(b)가 추가됨.
  - NiceGame(b)는 참이 아님.

(¬: classical negation, ~: negation as failure)

OWL을 넘어

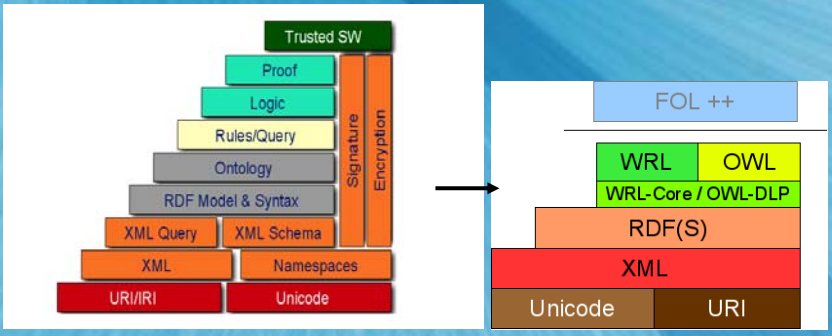
온톨로지 확장 (1)



- 언어
  - SWRL
  - WRL

OWL을 넘어
온톨로지 확장 (2)

- 새로운 시맨틱웹 케이크가 제안되기도.....



OWL을 넘어
온톨로지 확장 (3): WRL 문서의 예

```
namespace {_"http://www.example.org/ontologies/example#",
 dc     _"http://purl.org/dc/elements/1.1#",
 foaf  _"http://xmlns.com/foaf/0.1/",
 xsd    _"http://www.w3.org/2001/XMLSchema#",
 wrl  _"http://www.wsml.org/wsml/wrl-syntax#",
 loc    _"http://www.wsmo.org/ontologies/location#"}
```
Namespace Declaration

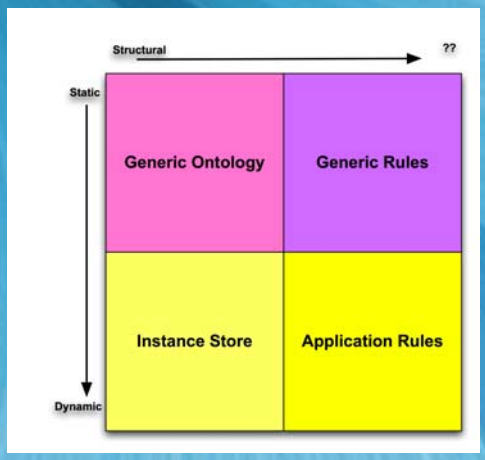
```
ontology Family
 nfp
  dc#title hasValue "WRL example ontology"
 endnfp
```
Ontology Annotation

```
concept Human subConceptOf { Primate, LegalAgent }
  nonFunctionalProperties
    dc#description hasValue "concept of a human being"
  endNonFunctionalProperties
  hasName ofType foaf#name
```
Concept Definition

```
relation ageOfHuman/2 (ofType Human, ofType _integer)
  nonFunctionalProperties
    dc#relation hasValue FunctionalDependencyAge
  endNonFunctionalProperties
```
Property Definition

```
axiom FunctionalDependencyAge
  definedBy
    !- ageOfHuman(?x,?y1) and
       ageOfHuman(?x,?y2) and wrl#numericInequal(?y1,?y2).
```
Axiom(Rule) Definition

네 부류의 지식
구성



네 부류의 지식
예

- Generic Ontology
  - 차종 체계, 부품 체계
- Generic Rules
  - 배기량에 따른 차급 분류, 부품의 포함 관계
- Instance Store
  - 자동차 및 부품 인스턴스들 선언
- Application Rules
  - 고장난 부품의 종류, 차종 및 연식, 보험 사양 등에 따른 A/S 정책

네 부류의 지식

정책

- 대부분의 어플리케이션들은 네 부류의 지식을 모두 필요로 한다.
- 도메인을 구성하는 지식들을 적절히 분류하고 각 부류 별로 적합한 데이터 표현 기법을 적용/활용하여야 한다.
- 규칙과 온톨로지를 적절히 혼용하여야 효과적인 데이터 표현 및 처리가 가능하다.

OWL 추론 엔진

Description Logic 추론 부류

- 추론 알고리즘: Tableaux Algorithm
  - 비교 흡수 부정(Resolution Refutation)과 유사
- 이론적 타당성
  - Sound하고 Complete한 추론
- 대표 엔진
  - *FaCT, Racer, Pellet etc*

OWL 추론 엔진

메타 추론 부류 (1)

- OWL의 의미론을 해석할 수 있는 메타 추론 규칙을 통해 추론
  - 메타 추론 규칙은 FOL, F-Logic, Production Rules, Logic Programming 등 다양한 논리 체계를 통해 기술
- 이론적 타당성
  - Sound한 추론 가능. Complete한 추론은 불가능.
- 실용성 및 확장성
  - 비단조 추론 수용 가능
  - 함수 활용, 외부 객체 연동 등 실용 기능 수용 가능
  - 규칙과 온톨로지 결합 활용 가능

OWL 추론 엔진

메타 추론 부류 (2)

- 대표 엔진
  - 부류 1: Rule Engine
    - *OWLJessKB, Jena, Bossam*
  - *부류 2: FOL Reasoner*
    - *JTP, Hoolet*
  - *부류 3: Logic Programming*
    - *Euler, XSB Prolog*
  - 부류 4: F-Logic
    - F-OWL

OWL 추론 엔진

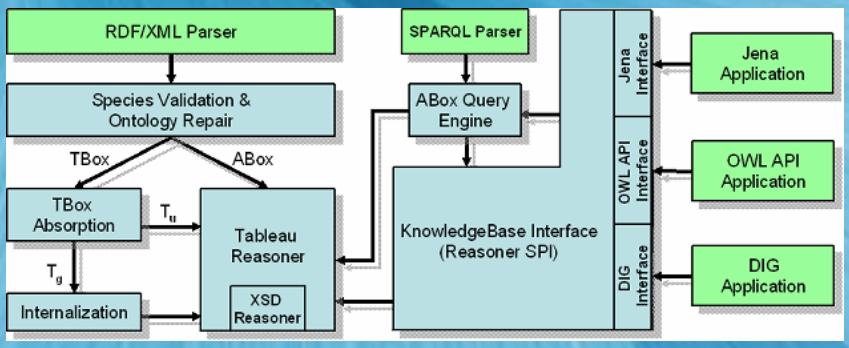
Pellet (1): 소개

- UMIACS*에서 개발한 오픈소스 Java 소프트웨어
- Sound & Complete OWL DL Reasoner
  - Tableaux 알고리즘 기반
  - 현재 DL 추론 엔진 중 가장 폭넓은 OWL 표현력을 지원
- 주요 추론 기능
  - Ontology Repair
  - Species Validation
  - Consistency Check
  - Classification
  - Entailments Check
  - Answering a subset of RDQL queries
- DIG 인터페이스 지원

*University of Maryland Institute for Advanced Computer Studies

OWL 추론 엔진

Pellet (2): 구조



OWL 추론 엔진

JENA (1): 소개

- HP에서 개발한 오픈소스 Java 소프트웨어
- 주요 이력
  - 2000년부터 개발. RDF 파서로 시작함.
  - 2003년 DAML+OIL 지원
  - 2004년 OWL 지원, 추론 기능 지원하기 시작.
- 다양한 기능 구성
  - ARP 이벤트 기반 RDF 파서
  - Database backend
  - RDQL 질의문 파싱 및 처리
  - Jena Inference API (Plug-in 구조)
  - DIG Interface
- 각종 추론 엔진을 포함
  - Transitive Reasoner, RDFS Reasoner, OWL, OWL Mini, OWL Micro Reasoners, DAML Micro Reasoner, Generic Reasoner

OWL 추론 엔진

JENA (2): RDFS 추론 엔진

- 대부분의 RDFS Entailment 계산 가능
- 세가지 작동 모드

  (지원되는 RDFS 의미론의 범위에 따름)
  - Full
  - Default
  - Simple
    - rdfs:subPropertyOf, rdfs:subClassOf, rdfs:domain, rdfs:range

OWL 추론 엔진

JENA (3): OWL 추론 엔진

- Sound but Incomplete Reasoning
- OWL Lite 까지 지원
- 세가지 추론 모드
  - OWL Micro Reasoner
    - RDFS + {intersectionOf, unionOf, hasValue}
  - OWL Mini Reasoner
    - OWL - {forward entailments from minCardinality & someValuesFrom}
  - OWL Reasoner
    - OWL Lite subset of OWL Full

OWL 추론 엔진

JENA (4): 규칙 엔진

- 전향 및 후향 추론 엔진을 모두 포함
  - RETE 기반 전향 추론 엔진
  - Datalog 기반 후향 추론 엔진
- 작동 모드
  - 전향(Forward Chaining) 및 후향(Backward Chaining)
  - 혼합(Hybrid Execution)
- 규칙 언어의 특징
  - Triple 모델에 기반한 술어 표현 (N-ary 술어 표현 불가능)
  - Negation 표현을 지원하지 않음

OWL 추론 엔진

JENA (5): 규칙 문법

```
Rule      :=  bare-rule .  or   [ bare-rule ] or   [ ruleName : bare-rule ]
bare-rule :=  term, ... term -> hterm, ... hterm     // forward rule
          or  term, ... term <- term, ... term       // backward rule
hterm     :=  term or   [ bare-rule ]
term      :=  (node, node, node)          // triple pattern
          or  (node, node, functor)       // extended triple pattern
          or  builtin(node, ... node)     // invoke procedural primitive
functor   :=  functorName(node, ... node)          // structured literal
node      :=  uri-ref                      // e.g. http://foo.com/eg
          or  prefix:localname             // e.g. rdf:type
          or  ?varname                     // variable
          or  'a literal'                  // either a string or a number
          or  number                       // e.g. 42 or 25.5
```


OWL 추론 엔진

JENA (6): 규칙 예

```
[allID: (?C rdf:type owl:Restriction), (?C owl:onProperty ?P),
    (?C owl:allValuesFrom ?D)  -> (?C owl:equivalentClass all(?P, ?D)) ]

[all2: (?C rdfs:subClassOf all(?P, ?D)) -> print('Rule for ', ?C)
          [all1b: (?Y rdf:type ?D) <- (?X ?P ?Y), (?X rdf:type ?C) ] ]

[max1: (?A rdf:type max(?P, 1)), (?A ?P ?B), (?A ?P ?C)
     -> (?B owl:sameAs ?C) ]
```

OWL 추론 엔진
보쌈 (1): 소개

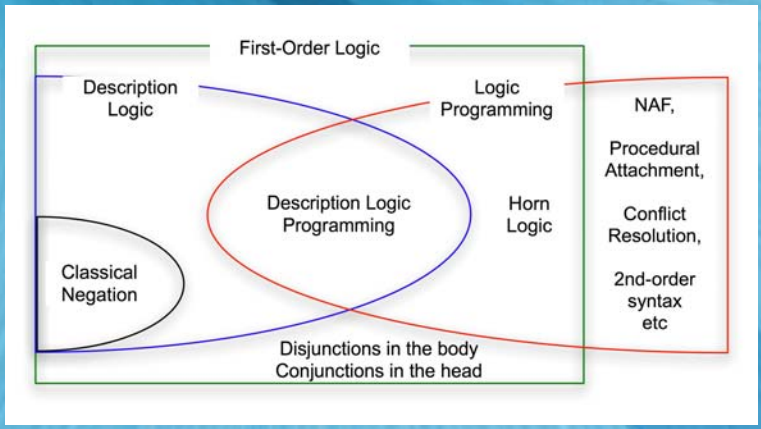
- 이력
  - 2003년에 개발 시작
  - 2003년 말에 초기 버전 개발 완료.
    - OWL Test Case를 이용하여 추론 기능 검증
  - 2004년 유지/보수
  - 2005년 v0.7, v0.8 개발
    - 바이너리 버전 공개 중
    - RuleML 및 SWRL 지원, 성능 최적화, Java 객체 연동 기능 확장, OWL 추론 기능 확장 등

OWL 추론 엔진
보쌈 (2): 주요 기능

- 표현력
  - 확장된 혼 논리(Horn Logic)
  - 프레임(Frame) 지식 표현
  - 고차 논리 술어 표현
  - NAF 및 Classical Negation 지원
  - 규칙 우선 순위 설정
- 규칙/온톨로지 추론 지원
  - RDFS, OWL, RuleML, SWRL 등 주요 웹 온톨로지 언어 및 웹 규칙 언어 지원
  - 온톨로지와 규칙을 결합 활용
- 실용성
  - 자바 객체 연동
  - 실행 API
  - Query 처리 (Buchingae Query만 지원)

OWL 추론 엔진

보쌈 (3): 표현력



OWL 추론 엔진

보쌈 (4): 규칙 기술 예

```
prefix builtin = http://www.etri.re.kr/2003/10/bossam-builtin#;
prefix family = http://www.etri.re.kr/Family#;
namespace is http://www.etri.re.kr/samples#;
rulebase rb01
{
  fact F01 is family:hasFather(Sam,John);
  fact F02 is family:hasFather(John,Jim);
  rule R01 is
    if
      family:hasFather(?x,?y) and family:hasFather(?y,?z)
    then
      family:hasGrandFather(?x,?z)
      and builtin:print(?x, " is the father of ", ?z);
}
```

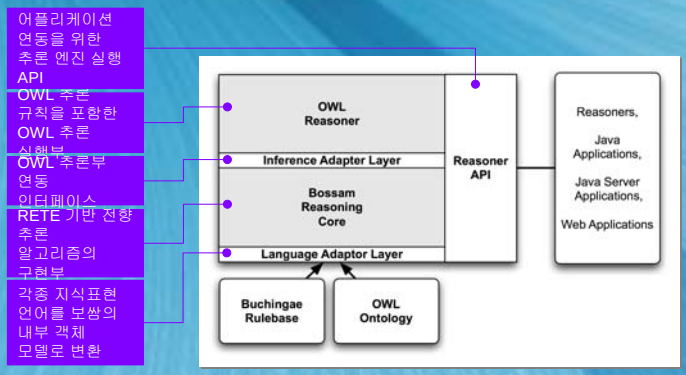
OWL 추론 엔진

보쌈 (5): OWL 추론 성능

- 어휘 지원 범위
  - OWL의 모든 어휘
- 완전성
  - Sound
  - Incomplete
- 성능
  - RETE 알고리즘 덕분으로 양호한 성능
    - W3C Wine 온톨로지: 로딩 및 추론에 약 80~90초 소요.
  - Jena 처럼 지원 어휘 축소를 통해 성능 향상 가능

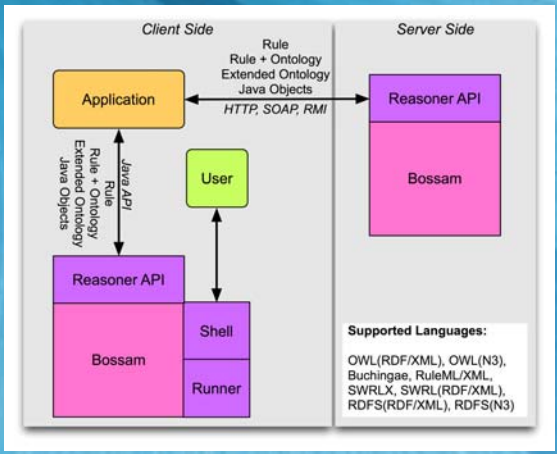
OWL 추론 엔진

보쌈 (6): 시스템 구조



OWL 추론 엔진

보쌈 (7): 활용 구조



OWL 추론 엔진

보쌈 (8): 데모

- Wine 온톨로지 로딩
- 질의/응답
  - 속성을 지정하여 와인 찾기.
  - 질의: hasFlavor(Dry) & hasColor(Red) & hasBody(Full) &...
- OWL 추론 실행
  - 전향 추론을 통해 유도된 모든 사실들이 출력됨.
  - 현재 약 14000여 개의 사실들이 유도됨.
- 질의/응답
  - 속성을 지정하여 와인 찾기.
  - 질의: hasFlavor(Dry) & hasColor(Red) & hasBody(Full) &...

참고문서

[1] W3C, OWL Overview
[2] W3C, OWL Reference
[3] W3C, OWL Abstract Syntax & Semantics
[4] W3C, OWL Guide
[5] Jos de Bruijin, “Lecture 7 - Description Logic Reasoning”, May 8, 2005.
[6] HP, “Jena 2 Inference Support”
[7] Jos de Brujin et al., “Web Rule Language”
[8] Sean Bechhofer, “OWL and Inference: Practical Examples”
[9] Pellet OWL Reasoner (http://www.mindswap.org/2003/pellet/)