逻辑、本体、推理以及实战工具与方法

王丛

文因互联 - Memect

2016 年 6 月 17 日

目录

MEMECT 文因互联

Classic Logic

Computational Logic is trying to solve:

- how to formulate the model
- how to compute the model

Computational Logic has 3 components:

- a well-defined *syntax*
- a well-defined *semantics*
- a well-defined *proof-theory*

Propositional Logic - Syntax

Example: $\neg A \wedge B \vee C \rightarrow X \wedge \neg Y$

An atomic formula is a propositional variable. Formulas are defined by the following inductive process.

- All atomic formulas are formulas.
- For every formula F, $\neg F$ is a formula.
- For all formulas F and G, also $(F \vee G)$ and $(F \wedge G)$ are formulas.

Propositional Logic - Semantics

Define a function $\mathcal{A} : \mathcal{A}(F) \rightarrow \{0, 1\}$, where F is a formula, and $\{0, 1\}$ represents True and False.

- $\mathcal{A}(F \wedge G) = 1$ if $\mathcal{A}(F) = 1$ and $\mathcal{A}(G) = 1$, otherwise 0
- $\mathcal{A}(F \vee G) = 1$ if $\mathcal{A}(F) = 1$ or $\mathcal{A}(G) = 1$, otherwise 0
- $\mathcal{A}(\neg F) = 1$ if $\mathcal{A}(F) = 0$, otherwise 0

Propositional Logic - Proof

- Truth table
- Tableau Algorithm (Top-Down Tree)
- Resolution Algorithm (Bottom-Up Tree)

Tableau - Unsatisfiable

Given $\{(a \vee \neg b) \wedge b, \neg a\}$. Every branch is closed (conflicted).

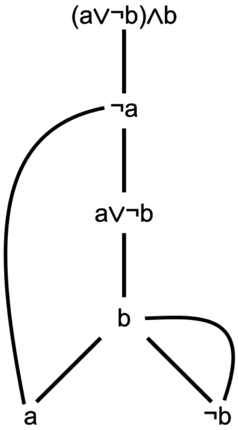
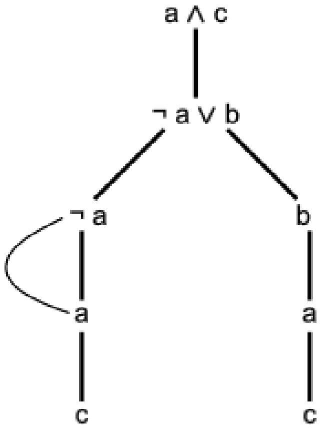


Tableau - Satisfiable

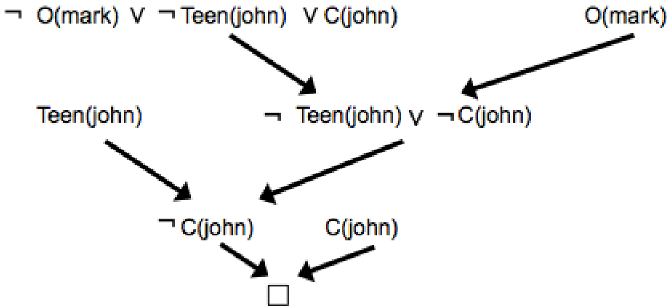
Given $\{(a \wedge c) \wedge (\neg a \vee b)\}$. Exist a branch is not closed.



Resolution

- Normalize all statement into Disjunction Normal Formal (DNF)
- Inferring between two DNF clause, one has a positive elment, the other has the negation of it.
- Inferring until no new clause can be inferred.

Resolution - Cont.



First-Order Logic

Example :

- Every cat is cute or extremely cute.
  $\forall x : Cat(x) \rightarrow Cute(x) \vee ECute(x)$
- Every cat has someone who loves it.
  $\forall x : Cat(x) \rightarrow \exists y : People(y) \vee Loves(y, x)$

First-Order Logic - Syntax

- $P(x_1, \ldots, x_k)$ is a formula.
- For each formula F, $\neg F$ is a formula.
- For all formulas F and G, $F \wedge G$ and $F \vee G$ are formulas.
- If x is a variable and F is a formula, $\exists x F$ and $\forall x F$ are formulas.

First-Order Logic - Semantics

Define U as a space, $\mathcal{A}$ as a function where mapping variables x in predicate P to some elements on the space, and mapping predicate P to a k-ary relation on the space.

- $\mathcal{A}(F(x_1, \ldots, x_k)) = 1$ if $(\mathcal{A}(x_1), \ldots, \mathcal{A}(x_k)) \in F^{\mathcal{A}}$, otherwise 0.
- $\mathcal{A}(F \wedge G) = 1$ if $\mathcal{A}(F) = 1$ and $\mathcal{A}(G) = 1$, otherwise 0.
- $\mathcal{A}(F \vee G) = 1$ if $\mathcal{A}(F) = 1$ or $\mathcal{A}(G) = 1$, otherwise 0.
- $\mathcal{A}(\neg F) = 1$ if $\mathcal{A}(F) = 0$, otherwise 0.
- $\mathcal{A}(\forall x F) = 1$ if for all $x \in U_{\mathcal{A}}$, $\mathcal{A}(F(x)) = 1$, otherwise 0
- $\mathcal{A}(]\exists x F) = 1$ if exists some $x \in U_{\mathcal{A}}$, $\mathcal{A}(F(x)) = 1$, otherwise 0

MEMECT 文因互联

First-Order Logic - Semantics

$$F = \forall x \forall y (P(a) \wedge (P(x) \rightarrow Q(x, x)))$$

Example1: $U_{\mathcal{A}} = \mathcal{N}, P^{\mathcal{A}} = \mathcal{N}, Q^{\mathcal{A}} = \{(n, k) | n < k\}$

Example2: $U_{\mathcal{A}} = \{☺, ☹\},\ P^{\mathcal{A}} = U_{\mathcal{A}},\ Q_{\mathcal{A}} = \{(☹, ☹)\}$

First-Order Logic - Proof

The problem “Given a First-formula F, is F valid?”is undecidable.
(proof by reduction of the Halting Problem.)

Reasoning Algorithms:

- Tableau
- Resolution

Monotonic Logic

The conclusion is not changed by the addition of premises.

- If $K \models F$, then $K \sqcup G \models F$.
- If $M \subseteq N$, then $\{F | M \models F\} \subseteq \{F | N \models F\}$.

Non-monotonic Logic

If $K \models F$, then not necessarily $K \sqcup G \models F$.

- Typically birds fly.
- Penguins do not fly.
- Tweety is a bird.
- Tweety flies.

Non-monotonic Logic

If $K \models F$, then not necessarily $K \sqcup G \models F$.

- Typically birds fly.
- Penguins do not fly.
- Tweety is a bird.
- Tweety flies.

Non-monotonic Logic

If $K \models F$, then not necessarily $K \sqcup G \models F$.

- Typically birds fly.
- Penguins do not fly.
- Tweety is a bird.
- Tweety flies.
- Tweety is a penguin.

Non-monotonic Logic

If $K \models F$, then not necessarily $K \sqcup G \models F$.

- Typically birds fly.
- Penguins do not fly.
- Tweety is a bird.
- Tweety flies.
- Tweety is a penguin.

Non-monotonic Logic

If $K \models F$, then not necessarily $K \sqcup G \models F$.

- Typically birds fly.
- Penguins do not fly.
- Tweety is a bird.
- ~~Tweety flies.~~
- Tweety is a penguin.

The previous conclusion must be retracted, such that Tweety does not fly will hold.

Non-monotonic History

Non-Monotonic logics have been proposed at the beginning of the 80's, here are historically the most important proposals.

- Non-monotonic logic, by McDermott and Doyle, '80
- Default Logic, by Reiter, '80
- Circumscription, by McCarthy, '80
- Autoepistemic logic, Moore, '84

Most of current research is based on these 3 main frames.

Frame Problem

Problem: How to represent that objects are not affected by state change?

Example: Moving an object does not change its color.

- $color(x, c, s) \rightarrow color(x, c, result(move, x))$
- $color(x, c, s) \rightarrow color(x, c, result(open\_door, x))$
- $color(x, c, s) \rightarrow color(x, c, result(a\_cat\_pass\_by, x))$

We need a great number of frame axioms

Frame Problem

Problem: How to represent that objects are not affected by state change?

Example: Moving an object does not change its color.

- $color(x, c, s) \rightarrow color(x, c, result(move, x))$
- $color(x, c, s) \rightarrow color(x, c, result(open\_door, x)$
- $color(x, c, s) \rightarrow color(x, c, result(a\_cat\_pass\_by, x)$

We need a great number of frame axioms

Frame Problem

We nee a *General axiom* of such form:

$holds(p, s) \wedge \neg exception(p, a, s) \rightarrow holds(p, result(a, s))$

But what action is not an exception?

We need a non-monotonic reasoning mechanism.

Frame Problem

We nee a *General axiom* of such form:

$holds(p, s) \wedge \neg exception(p, a, s) \rightarrow holds(p, result(a, s))$

But what action is not an exception?

We need a non-monotonic reasoning mechanism.

CWA and OWA

- OWA (Open World Assumption):

  Define: What is True only if it's known to be True

  Used in Semantic Web, assume no one has complete knowledge.

  Monotonic Logic.

CWA and OWA

- OWA (Open World Assumption):
  Define: What is True only if it's known to be True
  Used in Semantic Web, assume no one has complete knowledge.
  Monotonic Logic.
- CWA (Closed World Assumption):
  Define: What is not known to be True, is False.
  Used in Database System, whhich is assumed to be complete.
  Also called Negation as failure.

CWA and OWA

- OWA (Open World Assumption):
  Define: What is True only if it's known to be True
  Used in Semantic Web, assume no one has complete knowledge.
  Monotonic Logic.
- CWA (Closed World Assumption):
  Define: What is not known to be True, is False.
  Used in Database System, whhich is assumed to be complete.
  Also called Negation as failure.
  Non-monotonic Logic.

CWA and OWA

- OWA (Open World Assumption):
  Define: What is True only if it's known to be True
  Used in Semantic Web, assume no one has complete knowledge.
  Monotonic Logic.
- CWA (Closed World Assumption):
  Define: What is not known to be True, is False.
  Used in Database System, whhich is assumed to be complete.
  Also called Negation as failure.
  Non-monotonic Logic.

本体

本体是定义了关系 (semantics) 的一组领域词汇 (entities)

- **TBox** 类的关系 (subClassOf, objectProperty, etc.)
- **RBox** 关系的关系 (Property Chains, etc.)
- **ABox** 实例的关系 (typeOf, etc.)

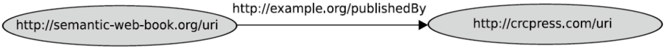
本体语言

- **RDF** 实例的关系
- **RDFS** 类的关系
- **OWL** 基于描述逻辑 (DL) 的更加复杂的类关系
  - **OWL EL** Modelling 医学、法律本体
  - **OWL RL** Query
  - **OWL QL** Rule

RDF

RDF idea

- ❑ **Use (directed) graphs as data model**



Subject **Predicate** **Object**

- ❑ **“Resource Description Framework”**

RDFS

Classes and Instances

- Classes stand for sets of things.
  In RDF: Sets of URIs.
- book:uri is a member of the class ex:Textbook

```
book:uri    rdf:type    ex:Textbook .
```

- a URI can belong to several classes

```
book:uri    rdf:type    ex:Textbook .
book:uri    rdf:type    ex:WorthReading .
```

- classes can be arranged in hierarchies: each textbook is a book

```
ex:Textbook    rdfs:subClassOf    ex:Book .
```

RDFS cont.

Implicit knowledge

- if an RDFS document contains

```
u    rdf:type    ex:Textbook .
```

and

```
ex:Textbook    rdfs:subClassOf    ex:Book .
```

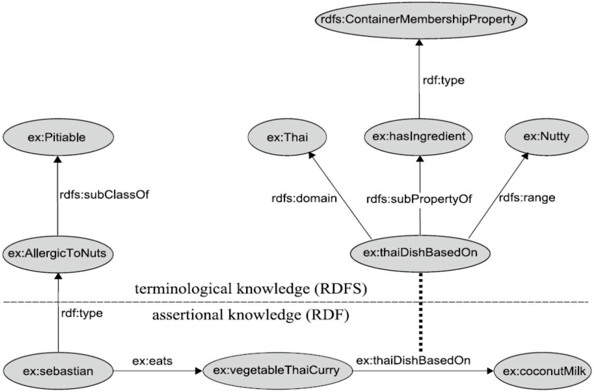
then

```
u    rdf:type    ex:Book .
```

is *implicitly* also the case: it's a *logical consequence*. (We can also say it is *deduced* (deduction) or *inferred* (inference)

RDF - RDFS

The same as graph



本体语言

- **RDF** 实例的关系
- **RDFS** 类的关系
- **OWL** 基于描述逻辑 (DL) 的更加复杂的类关系
  - **OWL EL** Modelling 医学、法律本体
  - **OWL RL** Query
  - **OWL QL** Rule

描述逻辑

描述逻辑是一阶逻辑 (FOL) 的可判定子集，W3C 标准 OWL 语言的逻辑基础

- 类 (Concept or Class)
  一元关系：猫，公司
- 关系 (Role or Property)
  二元关系：喜欢，雇佣
- 实例 (Individual)
  常量：咪咪，文因互联

描述逻辑语法[1]

- TBox

  科技公司 $\sqsubseteq$ 公司

  公司 $\sqsubseteq \exists$hasPresident.CEO
- RBox

  hasFather $\circ$ hasFather $\sqsubseteq$ hasGrandFather
- ABox

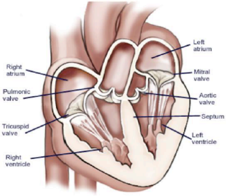
  CEO(鲍捷)

[1] http://dblp.uni-trier.de/db/conf/dlog/handbook2003.html

描述逻辑 - 举例

Example

Heart is a muscular organ that
is part of the circulatory system



$$Heart \sqsubseteq MuscularOrgan \sqcap \exists part\text{-}of.CirculatorySystem$$

描述逻辑复杂度

- OWL **NExpTime-complete**
- OWL 2 **2NExpTime-complete**
- OWL 2 EL **PTime-complete**
- OWL 2 RL **PTime-complete**
- OWL 2 QL $\mathbf{AC}^0$

描述逻辑与规则的转换

- TBox

  TechCompany(x) $\rightarrow$ Company(x)

  Company(x) $\rightarrow$ hasPresident(x,y) $\wedge$ CEO(y)

- RBox

  hasFather(x, y) $\wedge$ hasFather(y,z) $\rightarrow$ hasGrandFather(y,z)

- ABox

  CEO(鲍捷)

OWL 本体推理

常用推理机[2]

推理机	ELK	RDFox	TrOWL	Konclude	Pellet
算法	Completion Rule	Datalog	Approximation	Tableau	Tableau
Soundness	√	√	√	√	√
Completeness	√	√	×	√	√
针对 OWL 语言	OWL EL	OWL RL	OWL 2	OWL 2	OWL 2
特性	Best of EL	Best of RL	快，不完备	Best of OWL 2	慢，功能全
实现语言	Java	C	Java	C++	Java

[2] http://owl.cs.manchester.ac.uk/tools/list-of-reasoners/

推理机

OWL2 推理机

- Pellet
- Hermit
- TRon
- MoRe
- Konclude

OWLEL、RL 推理机

- ELK
- RDFox

本体解析工具

用于解析 RDF, OWL

- Jena - Java 语言，支持简单的存储与查询 (rdb,tdb)，支持简单的推理 (Rule)
- OWLApi - Java 语言，纯解析工具，最适合解析 OWL，大部分推理机再其上开发的
- rdflib - Python 语言，纯解析工具，支持一些 Sparql 的接口，

Neo4j

优点:

- 支持 Relation Property，避免 Reification
- 良好的 Api 以及社区支持
- 商业版支持分布式
- 易上手

缺点: 各种坑

- 插入需要写 batch transaction，否则极慢
- 删除数据很坑，要不直接删 graph.db 文件，要不用 batch 的方式先遍历再删除
- Optional Match 一多，查询极慢，需要拆开处理
- 不支持推理，Query 写法不够自由

Stardog

优点:

- 支持推理
- 支持一些特别规则的写法
- 支持 Validating Constraints

缺点:

- 标准的 Triple Store，学习成本较高
- 数据量大后，查询速度慢，推理速度慢

Stardog

- Titan - 支持分布式查询，查询速度快，但近 1 年没人维护
- Graphx - 据说很快，但目前刚起步
- Postgres - 干啥都行，稳定，坑少，但做图查询很繁琐

明确核心推理任务

- TBox or ABox
- Classification or Instance Retrieval
- Precision or Recall (2-8 原则)
- 复杂% or 简单%

选择推理机

- 复杂度是多少[3]
- 本体与规则是否可以转化 (RDfox or EL)
- 本体是否可以拆分 (Modular-based Reasoner[4])
- 任务是否可以拆分 (Pipeline Reasoning)

[3] http://www.cs.man.ac.uk/~Eezolin/dl/

[4] https://code.google.com/p/more-reasoner/

简化 OWL 复杂度

- 简化 inverse role 和 universal role[5]
- 简化 Role Transtivity[6]
- 简化 Datatype
- 小心使用 Class Equivalence
- 小心使用 Individual as Class

[5] http://120.52.72.44/daselab.cs.wright.edu/c3pr90ntcsf0/pub/ijcar2014.pdf

[6] http://www.cs.ox.ac.uk/boris.motik/pubs/motik06PhD.pdf

Query as Reasoning

- 直接将推理用查询展开 (OBDA[7], stardog[8])
  $\mathcal{T} = \{A \sqsubseteq B; B \sqsubseteq C\}$
  Query: C(?x)
  Rewrite Query: A(?x) ∪ B(?x) ∪ C(?x)
- 使用传统数据库代替

[7] http://ontop.inf.unibz.it
[8] http://stardog.com

Query in Neo4j

OPTIONAL MATCH (s:)-[:r1]->(o)
OPTIONAL MATCH (s)-[:r1]->()-[:r2]->(o)
OPTIONAL MATCH (s)-[:r2]->()-[:r2]->(o)
OPTIONAL MATCH (s)-[r2]->()-[r1]->()-[r2]->(o)
OPTIONAL MATCH ...
RETURN .....

Query in Sparql

```
Select ?s, ?o
Where {
?s (:r1|:r2)+ ?o .
}
```

Thanks

因为是内部讲义，所以部分内容未严格定义

参考 1 http://www.lsis.org/olivetti/TEACHING/MASTER/INTRONMR1.pdf

参考 2 http://www.computational-logic.org/content/events/iccl-ss-2005/lectures/bonatti/iccl-slides.pdf

参考 3 http://arxiv.org/pdf/1201.4089v3.pdf